# SPECTRAL ESTIMATION FOR GRAPH SIGNALS USING REED-SOLOMON DECODING

A Thesis

by

ABHISHEK DEB

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

| | |
|---|---|
| Chair of Committee, | Krishna R.Narayanan |
| Co-Chair of Committee, | Scott Miller |
| Committee Members, | Aydin Karsilayan |
| | Rabi N. Mahapatra |
| Head of Department, | Miroslav M. Begovic |

August 2017

Major Subject: Electrical Engineering

# ABSTRACT

Spectral estimation, coding theory and compressed sensing are three important subfields of signal processing and information theory. Although these fields developed fairly independently, several important connections between them have been identified. One notable connection between Reed-Solomon(RS) decoding, spectral estimation, and Prony's method of curve fitting was observed by Wolf in 1967. With the recent developments in the area of Graph Signal Processing(GSP), where the signals of interest have high dimensional and irregular structure, a natural and important question to consider is can these connections be extended to spectral estimation for graph signals?

Recently, it has been shown that a bandlimited graph signal that is $k$-sparse in the Graph Fourier Transform (GFT) domain can be reconstructed from $2k$ measurements obtained using a dynamic sampling strategy. Inspired by this work, we establish a connection between coding theory and GSP to propose a sparse recovery algorithm for graph signals using methods similar to Berlekamp-Massey algorithm and Forney's algorithm for decoding RS codes. In other words, we develop an equivalent of RS decoding for graph signals. The time complexity of the recovery algorithm is $O(k^2)$ which is independent of the number of nodes $N$ in the graph. The proposed framework has applications in infrastructure networks like communication networks, power grids etc., which involves maximization of the power efficiency of a multiple access communication channel and anomaly detection in sensor networks.

DEDICATION

*To my wonderful parents, Pronoti Deb and Debashis Deb, for their unconditional love and support.*

ACKNOWLEDGMENTS

I have been fortunate to have an advisor like Dr. Krishna R. Narayanan, who has been instrumental in motivating me throughout my thesis. I extend my gratitude to him for providing important professional advise that has enhanced my graduate experience. I also deeply appreciate his effort to give me an opportunity to present my work in the Graph Signal Processing Workshop held at Carnegie-Mellon University, Pittsburgh. The experience was amazing and exposed me to a lot of different researchers working in a variety of research problems.

Next, I would like to thank all my committee members for taking the time and effort to conduct my defense and providing useful insights to improve my thesis work. I would next like to thank my research group mate Nagaraj Thenkarai Janakiraman for continuously guiding me throughout my thesis work. His inputs have been instrumental in the progress and completion of my thesis. I also extend my thanks to the other members of the research group like Kiran Kumar, Avinash Vem and Arman Hasanzadeh for helping me out whenever I was stuck in my research work.

I would next like to express my sincere gratitude to my undergraduate research advisor Dr. Asutosh Kar from the International Institute of Information Technology, India. My decision to pursue graduate studies was largely influenced by him. Without his guidance and encouragement, I would have never been able to pursue my MS in a wonderful institute like Texas A&M University.

A graduate experience without great friends is incomplete. I have been able to make great friends, involve in productive discussions and build wonderful memories throughout my graduate school experience. Specifically, I have had my best times with Amarnath Mahadevuni, Abhijeet Sahu and Jaybharath Allu, who have been great roommates for two

years. I would also like to thank Sreyoshi Patra, who has been a special person in my life and supported me throughout this journey. I would also like to extend my gratitude to my best friends Pranab Das, Bhabesh Senapati and Ambika Prasad, who have always been there for me in my good and bad times.

I deeply thank the university and department committees for having admitted me into the Master of Science (M.S.) program of the Department of Electrical and Computer Engineering and providing a great international exposure. Ultimately, I thank my parents, my family and the Almighty for their blessings and continuous support.

# CONTRIBUTORS AND FUNDING SOURCES

## Contributors

This work was supported by a thesis committee consisting of Professors Krishna R. Narayanan (chair), Scott Miller (co-chair) and Aydin Karsilayan of the Department of Electrical Engineering and Professor Rabi N. Mahapatra of the Department of Computer Science and Engineering. All other work conducted for the thesis was completed by the student independently.

## Funding Sources

This research work was not supported by any funding source.

# NOMENCLATURE

| | |
|---|---|
| DSP | Digital Signal Processing |
| GSP | Graph Signal Processing |
| GFT | Graph Fourier Transform |
| DFT | Discrete Fourier Transform |
| CS | Compressed Sensing |
| LASSO | Least Absolute Shrinkage and Selection Operator |
| OMP | Orthogonal Matching Pursuit |
| TV | Total Variation |
| MAC | Multiple Access Communication |
| TDMA | Time Division Multiple Access |
| SNR | Signal to Noise Ratio |

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

# 1. INTRODUCTION

Traditional Digital Signal Processing(DSP) rose to prominence about 50 years ago mainly due to the development of digital computer technology. The importance of DSP in every day life hardly needs to be overstated. The field of coding theory came into existence in the year 1948 when Shannon published his landmark paper *'A mathematical theory of communication'*. The area of spectral estimation rose to prominence in the 1960's when Tukey published a book *'An introduction to the frequency analysis of time series'*. Similarly, the area of compressed sensing has risen to prominence in the last decade. Even though these fields have emerged somewhat independently, there are indeed close connections between them.

Graph Signal Processing (GSP) is a very recent development in the signal processing community that seeks to extend notions of DSP to signals that are defined on graphs[1]. This is an exciting and nascent research field and the main theoretical principles as well as its applications to various areas are being studied. It is then natural to ask whether there are close connections between GSP, coding theory and compressed sensing. In this thesis, we explore one such a connection between Reed-Solomon decoding and spectral estimation for graph signals.

## 1.1 Need for GSP

A time series signal is a one-dimensional sequence of data points taken at successive equally spaced points in time. For processing such time series signals, several classical techniques such as the Fourier transform, filtering, compression etc., have been developed. In this age of big data, the volume and dimension of data is increasing and such data cannot be adequately modeled as time series signals. For example, data from sensor networks, social networks, transportation networks, biological networks have a rich structure and

cannot be modeled as a time series signal. For processing these complex data sets, several new approaches have been developed [2, 3, 4, 5]. One of the possible approaches is to represent this data as signals defined on graphs instead of signals defined on equally spaced points in time. Such signals are called graph signals and a natural question is to ask whether classical signal processing techniques can be applied for processing graph signals. Such an attempt to extend classical DSP to graph signals has resulted in the field of graph signal processing (GSP).

## 1.2 Error-correction coding

Error-correction coding also known as channel coding is a technique used especially in the field of telecommunication and data storage to control errors that occur during the transmission of data over noisy and untrusted communication channels. In this technique, the sender encodes a message of length $l$ i.e., $m = [m_1, m_2, ..., m_l]^T$ by introducing redundancy with the help of an error-correcting code. The resulting encoded message of length $n$ i.e., $c = [c_1, c_2, ..., c_n]^T$ is transmitted through a noisy channel which introduces errors in $c$. The noisy codeword is received by the receiver as $r = c + e = [r_1, r_2, ..., r_n]^T$ and decoded using efficient decoding algorithms to find the errors and retrieve the corrected message. A simple error-correction schematic and an example are shown in Fig. 1.1 and Fig. 1.2 respectively. The example shows redundancy being added to the message by the encoder and the message being retrieved from the noisy codeword by the decoder.

## 1.3 Spectral Estimation and Compressed Sensing

Spectral estimation of time series signals deals with estimating the distribution of total power of the signal over frequency from a finite record of time series signals. A classic result from spectral estimation that is relevant to this thesis that if the signal is sparse in the frequency domain, then it suffices to observe a small number of samples of the signal in the time domain in order to estimate the power spectrum of the signal. Further, there are

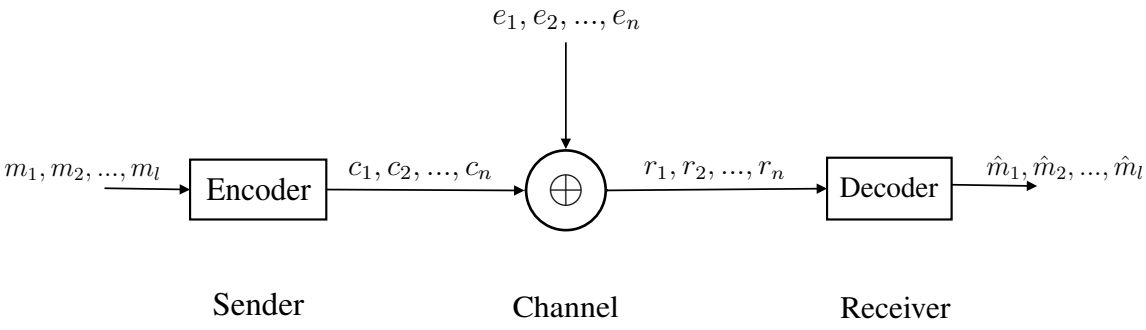



Figure 1.1: Schematic showing an encoder and decoder

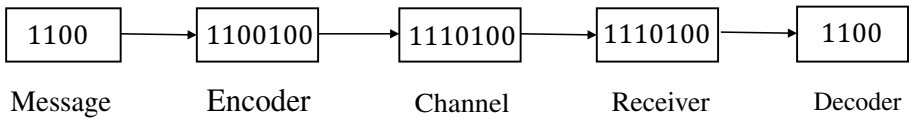



Figure 1.2: Example of channel coding

close connections between algorithms used to estimate the power spectrum and algorithm used for recovering errors in coding theory and algorithms for recovering sparse signals in compressed sensing.

Compressed sensing is a technique in signal processing that deals with efficient acquisition and reconstruction of a signal that is sparse in some domain. It shows a way to recover the signal with the help of fewer samples than required by the Shannon-Nyquist sampling theorem provided certain conditions are satisfied. Suppose, $x$ is a $k$-sparse vector over $\mathbb{R}$ or $\mathbb{C}$, then we can compress $x$ by storing only $y = \mathbf{A}x$. Here $y$ is a low-dimensional vector than $x$ and $\mathbf{A}$ is called the sensing matrix. The reconstruction involves finding the sparsest $x$ that satisfies $y = \mathbf{A}x$.

## 1.4 Problem Statement

Most of the work in spectrum estimation has been developed with the assumption that the signals of interest are time series signals or images. The main objective of this thesis is to extend some of this work to graph signals and to develop an algorithm for estimating the spectrum of graph signals that are sparse in the graph Fourier domain. We use techniques popular in coding theory and compressed sensing for the recovery of signals with a sparse graph spectrum. Specifically, we develop a Reed-Solomon style decoding algorithm for recovery of signals that are sparse in the graph Fourier transform domain. We also show that such algorithms have applications in sensor networks and in anomaly detection.

## 1.5 Outline of the thesis

Chapter 2 introduces coding theory and discusses decoding techniques of an important class of codes called RS codes. Chapter 3 discusses compressed sensing. Chapter 4 deals with spectrum estimation and Prony's method. Chapter 5 introduces the the framework of GSP and also discusses various concepts analogous to the classical DSP. A low complex spectral estimation technique for graph signals is proposed in Chapter 6. Chapter 7 and 8 explore some possible real world applications. Chapter 9 concludes the thesis and discusses potential future works in this area.

# 2. REED-SOLOMON CODES

## 2.1 Introduction

Mathematically, an $(n, k)$ block code $\mathcal{C}$ over a finite field $\mathbb{F}_q$ is a set of $q^k$ vectors of length $n$ called codewords. A linear block code refers to a block code for which any linear combination of codewords is also a codeword. It acts on a block of $k$ symbols from $\mathbb{F}_q$ to produce $n$ symbols of output data where the rest of the $(n-k)$ symbols are redundant. Some examples of block codes include Hamming codes, Bose-Chaudhuri-Hocquenghem(BCH) codes, Reed-Solomon(RS) codes etc. The codewords are formed in the encoder using a generator matrix(**G**) given by,

$$c = x\mathbf{G} \tag{2.1}$$

where, $x$ is the $1 \times k$ input signal vector, $c$ is the $1 \times n$ codeword of the linear block code $\mathcal{C}$ and **G** is the $k \times n$ generator matrix. A parity check matrix **H** of a linear block code $\mathcal{C}$ can be defined such that $\mathcal{C} = \left\{c \in \mathbb{F}_q : \mathbf{H}c^T = 0\right\}$. The decoder has the knowledge of these matrices **G** and **H**. When the erroneous message is received i.e., $r = c + e$, the decoder computes the syndromes $y$ given by,

$$y = \mathbf{H}r^T = \mathbf{H}(c^T + e^T) = 0 + \mathbf{H}e^T = \mathbf{H}e^T. \tag{2.2}$$

The decoding algorithm determines $e$ and corrects the noisy message.

## 2.2 Reed-Solomon Codes

For positive integers $m$ and $t$, an ($n$,$k$) RS code has the following properties,

1. Number of message symbols per codeword: $k$

2. Total number of encoded symbols per codeword: $n = 2^m - 1$

3. Code rate: $k/n$

4. Number of parity symbols per codeword: $n-k = 2t$, where, $t$ is the error correcting ability

5. Minimum symbol Hamming weight per codeword: $d = 2t + 1$

If out of the total $n$ symbols exactly $t$ or less of them are received in error then RS codes have the property that if $t \leq (n-k)/2$, the correct message vector can be computed from the noisy codeword.

### 2.2.1 RS encoding

Let us represent message as a sequence of coefficients of a polynomial. Let $x$, $y$ be vectors of length $m$ whose values are taken from the finite field $\mathbb{F}_q$ where, $q = 2^m$ and $k$ information symbols correspond to the $y$ vectors of points $(y_0, x_0), (y_1, x_1), ..., (y_{k-1}, x_{k-1})$ where, $x_0, x_1, ..., x_{k-1}$ be any $k$ distinct values of $x$. Using these $k$-points, we compute a unique polynomial of the form,

$$f(x) = f_{k-1}x^{k-1} + f_{k-2}x^{k-2} + ... + f_1x + f_0 \tag{2.3}$$

which fits these points. Therefore,

$$f(x) = \sum_{i=1}^{k} f_{i-1}x^{i-1} \tag{2.4}$$

which can be written in matrix form as,

$$F = f\mathbf{G} \tag{2.5}$$

where, **G** is the generator matrix in the polynomial form and the dimensions of $F$, $f$, **G** are $1 \times n$, $1 \times k$ and $k \times n$ respectively. So, to encode $k$ information symbols, evaluate $f(x)$ at $(n-k)$ additional points. The codeword consists of the $k$ information symbols $[y_0, y_1, ..., y_{k-1}]$ plus the $y$ values(redundant symbols) of the $n-k$ additional points. In other words, the codeword $[y_0, y_1, ..., y_{n-1}]$ is obtained by evaluating the polynomial $f(x)$ at $n$ different points $x_0, x_1, ..., x_{n-1}$. The most important thing to note is the structure of the matrix **G**, which is nothing but a Vandermonde structure and given as,

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & . & . & . & 1 \\ x_0 & x_1 & . & . & . & x_{n-1} \\ x_0^2 & x_1^2 & . & . & . & x_{n-1}^2 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ x_0^{k-1} & x_1^{k-1} & . & . & . & x_{n-1}^{k-1} \end{bmatrix}. \tag{2.6}$$

We know that Discrete Fourier Transform(DFT) establishes a duality between the coefficients of polynomials and their values i.e., let $p(x)$ and $q(x)$ be two polynomials of degree less than $n$. If the values of $p(x)$ are the coefficients of $q(x)$, then up to a scalar factor , the values of $q(x)$ are the coefficients of $p(x)$, provided the values are taken at locations given by $x_j = \alpha^j$ where, $\alpha$ is the primitive $n$-th root of unity and $j = 0, 1..., n-1$. So, equation (2.4) can be evaluated as,

$$\begin{aligned} y_j = f(x_j) &= f_{k-1}x_j^{k-1} + f_{k-2}x_j^{k-2} + ... + f_0 \\ &= \sum_{l=0}^{k-1} f_l \alpha^{jl} \qquad \text{for } j = 0, 1..., n-1. \end{aligned} \tag{2.7}$$

By padding zeros such that $f_k$ to $f_n$ are zeros we get the inverse DFT($\mathcal{F}^{-1}$) of the vector $[f_0, f_1, ...f_{k-1}, 0, 0, ...0]$ as shown below,

$$y_j = \sum_{l=0}^{n-1} f_l \alpha^{jl} = \mathcal{F}^{-1}[f_0, f_1, ...f_{k-1}, 0, 0, ...0]. \tag{2.8}$$

So, $[y_0, y_1, ..., y_{n-1}]$ is the codeword that is encoded and sent.

### 2.2.2 RS decoding

Let us consider the received codeword as $\tilde{y} = [\tilde{y}_0, \tilde{y}_1, ..., \tilde{y}_{n-1}]$ such that

$$\tilde{y}_j = y_j + z_j$$

where, $z_j$ is the noise at $j$-th time instant. Let the received polynomial be,

$$\tilde{f}(x) = \tilde{y}_{n-1}x^{n-1} + \tilde{y}_{n-2}x^{n-2} + ... + \tilde{y}_1 x + \tilde{y}_0. \tag{2.9}$$

So,

$$\tilde{f}(x_j) = \tilde{y}_{n-1}x_j^{n-1} + \tilde{y}_{n-2}x_j^{n-2} + ... + \tilde{y}_1 x_j + \tilde{y}_0 = \sum_{l=0}^{n-1} \tilde{y}_l \alpha^{jl}. \tag{2.10}$$

Now,

$$\begin{aligned} \tilde{f}(x_j^{-1}) = \sum_{l=0}^{n-1} \tilde{y}_l \alpha^{-jl} &= \sum_{l=0}^{n-1} y_l \alpha^{-jl} + \sum_{l=0}^{n-1} z_l \alpha^{-jl} \\ &= \sum_{l=0}^{n-1} y_l \alpha^{-jl} + \sum_{l=0}^{n-1} z_l \alpha^{-jl} \\ &= \mathcal{F}[y_0, y_1, ..., y_{n-1}] + \sum_{l=0}^{n-1} z_l \alpha^{-jl}. \end{aligned} \tag{2.11}$$

We know that $\mathcal{F}[y_0, y_1, ..., y_{n-1}]$ is zero for $j = k, ..., n-1$. So, from $j = k, k+1, ..., n-1$, $\tilde{f}(x_j^{-1})$ depends only on noise. Let exactly $t$ elements of the noise be non-zero

i.e., $z_{\gamma_0}, z_{\gamma_1}, ..., z_{\gamma_t}$ at arbitrary positions and $t < (n-k)/2$. Therefore,

$$\tilde{f}(x_{n-j}^{-1}) = \sum_{l=0}^{n-1} \tilde{z}_l \alpha^{-(n-j)l} = \sum_{l=0}^{n-1} z_l \alpha^{-nl} \alpha^{jl} \qquad \text{for } j = 1, 2, ...n-k. \tag{2.12}$$

Substituting the value of $\alpha = e^{\frac{2\pi i}{n}}$ where, $i = \sqrt{-1}$ we get,

$$\begin{aligned} \tilde{f}(x_{n-j}^{-1}) &= \sum_{l=0}^{n-1} z_l e^{-2\pi inl/n} e^{2\pi ijl/n} \\ &= \sum_{l=0}^{n-1} z_l e^{2\pi ijl/n} \\ &= \sum_{l=0}^{t} z_{\gamma_l} e^{2\pi ij\gamma_l/n}. \end{aligned} \tag{2.13}$$

Now, let $\tilde{f}(x_{n-j}^{-1})$ be represented as $s_j$, then,

$$s_j = \sum_{l=0}^{t} z_{\gamma_l} (e^{2\pi i\gamma_l/n})^j. \tag{2.14}$$

Let $W_l = e^{2\pi i\gamma_l/n}$. We expand the equation (2.14) in matrix form we get,

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ . \\ . \\ s_{n-k} \end{bmatrix} = \begin{bmatrix} W_1 & W_2 & . & . & . & W_t \\ W_1^2 & W_2^2 & . & . & . & W_t^2 \\ W_1^3 & W_2^3 & . & . & . & W_t^3 \\ W_1^4 & W_2^4 & . & . & . & W_t^4 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ W_1^{n-k} & W_2^{n-k} & . & . & . & W_t^{n-k} \end{bmatrix} \begin{bmatrix} z_{\gamma_1} \\ 0 \\ z_{\gamma_3} \\ 0 \\ . \\ . \\ z_{\gamma_t} \end{bmatrix}. \tag{2.15}$$

That is,

$$s = \mathbf{H}z. \tag{2.16}$$

The important thing to note is that the matrix $\mathbf{H}$ in right hand side of the equation has a Vandermonde structure. This sequence of $s_j$'s are called as Syndromes. The goal of the decoder is to use these syndromes to estimate the position($\gamma_l$) and the values of the non-zero errors($z_{\gamma_l}$). For this, two famous algorithms called as Berlekamp-Massey(BM) algorithm and Forney's algorithm are used respectively. The position of errors are given by the BM algorithm while the values of the errors are given by the Forney's algorithm. A brief description of these algorithms is given in Chapter 6.

# 3. COMPRESSED SENSING

## 3.1 Description

Compressed Sensing(CS) is a field of signal processing which deals with recovering sparse high dimensional signals from low-dimensional measurements. Consider a real-valued, finite-length, discrete-time signal $x$, which can be viewed as an $n \times 1$ column vector in $\mathbb{R}^n$ with elements $x[i]$, $i = 1, 2, ..., n$. Then any signal in $\mathbb{R}^n$ can be represented in terms of an orthonormal basis of $n \times 1$ vectors $\{\psi_i\}_{i=1}^n$. Using the $n \times n$ basis matrix with the vectors $\{\psi_i\}$ as columns, a signal $x$ can be expressed as,

$$x = \sum_{i=1}^{n} s_i \psi_i.$$

In matrix form it is,

$$x = \mathbf{\Psi} s \tag{3.1}$$

where, $s$ is the $n \times 1$ column vector of the weighting coefficients $s_i = \Psi_i^T x$. The signal $x$ is $k$-sparse if it is a linear combination of only $k$ basis vectors i.e., only $k$ of the $s_i$ coefficients in equation are non-zero. The signal $x$ is said to be compressible if in the equation (3.1) the representation has only a few ($k << n$) large coefficients and many($n - k$) small coefficients. The signal is then 'sampled' with $m << n$ linear measurements to obtain the values,

$$y = \mathbf{\Phi} x \tag{3.2}$$

where, $y$ is collection of measurements in a $m \times 1$ vector and $\mathbf{\Phi}$ is a collection of vectors $\{\phi_i\}_{i=1}^n$ to form a $m \times n$ matrix. Then, by substituting the value of $x$ from equation (3.1),

$y$ can be written as,

$$y = \mathbf{\Phi}x = \mathbf{\Phi}\mathbf{\Psi}s = \mathbf{A}s \tag{3.3}$$

where, $\mathbf{A} = \mathbf{\Phi}\mathbf{\Psi}$ and is known as the sensing matrix. Therefore, the CS problem consists of designing a stable $\mathbf{A}$ such that the important information in any sparse signal is not affected by the dimensionality reduction from $n$ to $m$, and a reconstruction algorithm to recover $x$ from $y$.

### 3.2 Recovery algorithms

The theoretical limitations of CS gives the bound on minimal number of measurements required and a condition on the sensing matrix $\mathbf{A}$. The important thing is that if $\mathbf{A}$ is a totally positive matrix, then equation (3.3) allows us to reconstruct every $k$-sparse vector with only $m = 2k$ measurements[6]. An example of a totally positive matrix is a Vandermonde matrix. The reconstruction algorithm involves solving an $l$-1 minimization problem given by,

$$\begin{aligned} &\text{minimize} \quad ||x||_1 \\ &\text{subject to} \quad y = \mathbf{A}x. \end{aligned} \tag{3.4}$$

There are other algorithms that are used like Least Absolute Shrinkage and Selection Operator(LASSO)[7] and Orthogonal Matching Pursuit(OMP) algorithm[8]. These algorithms are robust to noise and we will use them in Chapter 8 for an application in which noise is present. Assuming, $\mathbf{A}$ as Vandermonde matrix, equation (3.3) can be compared to equation (2.16) where both $s$ and $z$ are sparse vectors and the sensing matrix $\mathbf{A}$ is same as the matrix $\mathbf{H}$. This is the connection between CS and RS decoding. Along with this connection, another important thing to conclude is that only $2k$ measurements are sufficient to recover a $k$-sparse signal if the matrix $\mathbf{A}$ or $\mathbf{H}$ is a Vandermonde matrix.

# 4. SPECTRAL ESTIMATION USING PRONY'S METHOD

A time-domain representation shows how a time series signal varies over time and a frequency-domain plot shows how much of the signal lies within each given frequency band over a range of frequencies. Fig. 4.1 and 4.2 show a time domain and frequency domain representation of a time series signal respectively. Any process that involves calculation of amplitudes, powers, or intensities versus frequency is called as spectrum estimation. The unknown amplitudes or powers in frequency domain can be calculated by periodically sampling points in the time domain. Furthermore, if the signal is sparse in the frequency domain, then we can sample very few points in the time domain.

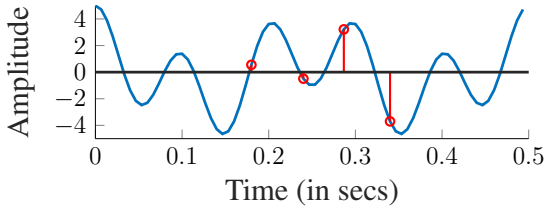



Figure 4.1: Time domain representation

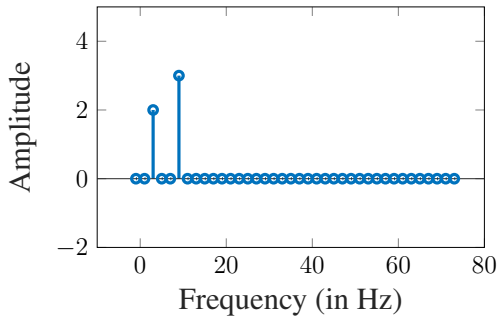



Figure 4.2: Frequency domain representation

We consider a curve fitting technique called as Prony's method to do spectral estimation. This method has a low computational complexity and a nice connection to RS decoding.

## 4.1 Prony's Method

Prony's Method was developed by Prony in 1795 in order to explain the expansion of various gases. It is a technique for modeling sampled data as a linear combination of exponentials. This method is not a spectral estimation technique but it has a very close relation to the least square linear prediction algorithms. It extracts complex exponential signals from time series signal by solving a set of linear equations. Assuming the $n$ complex time domain data samples $x_1, x_2..., x_n$, the function to be found is approximated by $t$ exponential functions as,

$$y_j = \sum_{l=1}^{t} A_l e^{(\alpha_l + i\omega_l)(j-1)T_p + i\psi_l} \tag{4.1}$$

where, $j = 1, 2, ..., n$, $T_p$ is the sampling period, $A_l$ is the amplitude, $\alpha_l$ is the damping factor, $\omega_l$ is the angular velocity and $\psi_l$ is the initial phase. Assuming the initial phase and angular velocity to be zero, let $z_l = A_l$ and $W_l = e^{\alpha_k T_p}$, then this function may be compactly written as,

$$y_j = \sum_{l=1}^{t} z_l W_l^{j-1}. \tag{4.2}$$

The aim is to fit the $y_j$'s to the observed $x_j$'s, such that the squared error over $n$ data values is given as,

$$\Delta(y) = \sum_{j=1}^{n} |x_j - y_j|^2 = \sum_{j=1}^{n} \Big| x_j - \sum_{l=1}^{t} z_l W_l^{j-1} \Big|^2. \tag{4.3}$$

The minimization problem is then given by,

$$y = \underset{\tilde{y}}{\text{argmin}} \quad \Delta(\tilde{y}). \tag{4.4}$$

Solving this turns out to be a difficult non-linear problem. Prony's method gives a way to solve this using linear equations. The $t$ equations of (4.2) can be expressed in matrix form as,

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_t \end{bmatrix} = \begin{bmatrix} W_1^0 & W_2^0 & . & . & . & W_t^0 \\ W_1^1 & W_2^1 & . & . & . & W_t^1 \\ W_1^2 & W_2^2 & . & . & . & W_t^2 \\ W_1^3 & W_2^3 & . & . & . & W_t^3 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ W_1^{t-1} & W_2^{t-1} & . & . & . & W_t^{t-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ . \\ . \\ . \\ z_t \end{bmatrix}. \tag{4.5}$$

This equation is a set of linear equations that is to be solved for unknown amplitudes $z_l$'s. Prony proposed to define a polynomial that has the $W_l$ exponents as its roots i.e.,

$$f(w) = \prod_{l=1}^{t}(w - W_l) = (w - W_1)(w - W_2)...(w - W_t). \tag{4.6}$$

When expanded, the polynomial can also be expressed as sum,

$$f(w) = \sum_{l=0}^{t} \Lambda_l W^{t-l} = \Lambda_0 W^t + \Lambda_1 W^{t-1} + ... + \Lambda_{t-1} W + \Lambda_t. \tag{4.7}$$

If as many data samples are used as there are exponential parameters then,

$$x_j = \sum_{l=1}^{t} z_l W_l^{j-1}. \tag{4.8}$$

Shifting the index on equation (4.7) $j$ to $j-l$ and multiplying by $\Lambda_l$ we get,

$$\Lambda_l x_{j-l} = \Lambda_l \sum_{k=1}^{t} z_k W_k^{j-l-1} \tag{4.9}$$

which can be written as,

$$\sum_{l=0}^{t} \Lambda_l x_{j-l} = \sum_{k=1}^{t} z_k W_k^{j-t} \Big[ \sum_{l=0}^{t} \Lambda_l W_k^{t-l-1} \Big]. \tag{4.10}$$

The bracketed summation in (4.9) is a polynomial similar to (4.6) and is evaluated at each of its roots $W_l$ giving,

$$\sum_{l=0}^{t} \Lambda_l x_{j-l} = 0. \tag{4.11}$$

This equation can be solved for the coefficients $\Lambda$'s and then the roots can be calculated using (4.6). The damping factors and frequencies are found from the roots $\Lambda$'s. If the number of samples is $2t$ then we can find an exact fit and if the number of samples is greater than $2t$ then we can find least square solution to the equations given below,

$$\sum_{l=0}^{t} \Lambda_l x_{j-l} = \Delta_j. \tag{4.12}$$

## 4.2 Connection between Prony's method and RS decoding

If we observe equations (2.15) and (4.5), there is a striking resemblance to their structure. Both contain the product of a Vandermonde matrix times a sparse vector. This means we can use the Prony's method to the syndrome decoding problem in order to convert the non-linear equations to linear ones. Let the position of errors be contained in $P_l$ and the

values of error at these positions be $E_l$. The syndrome equation (2.15) is then written as,

$$y_j = \sum_{l=1}^{t} E_l P_l^j. \tag{4.13}$$

We use the BM algorithm to find the error-locating polynomial given as,

$$\Lambda(w) = \prod_{l=1}^{t}(1 - wP_l) = 1 + \Lambda_1 w + ... + \Lambda_t w^t. \tag{4.14}$$

The roots of this polynomial are given by $P_l^{-1}$. That implies,

$$\Lambda(P_l^{-1}) = 1 + \Lambda_1 P_l^{-1} + ... + \Lambda_t P_l^{-t} = 0. \tag{4.15}$$

Proceeding in the similar way as in the previous section and multiplying both sides by $E_l P_l^{j+t}$ we get,

$$\begin{aligned} E_l P_l^{j+t}\Lambda(P_l^{-1}) &= E_l P_l^{j+t} + E_l P_l^{j+t}\Lambda_1 P_l^{-1} + ... + E_l P_l^{j+t}\Lambda_t P_l^{-t} \\ &= E_l P_l^{j+t} + \Lambda_1 E_l P_l^{j+t-1} + ... + \Lambda_t E_l P_l^j = 0 \end{aligned}$$

which can be written as,

$$\sum_{l=1}^{t} E_l P_l^{j+t} + \Lambda_1 \sum_{l=1}^{t} E_l P_l^{j+t-1} + ... + \Lambda_t \sum_{l=1}^{t} E_l P_l^j = 0.$$

Substituting the value of $y_j$ we get,

$$\begin{aligned} & y_{j+t} + \Lambda_1 y_{j+t-1} + ... + \Lambda_t y_j = 0 \\ & \Rightarrow y_j\Lambda_t + y_{j+1}\Lambda_{t-1} + ... + y_{j+t-1}\Lambda_1 = -y_{j+t}. \end{aligned}$$

In matrix form,

$$\begin{bmatrix} -y_{t+1} \\ -y_{t+2} \\ . \\ . \\ . \\ -y_{t+t} \end{bmatrix} = \begin{bmatrix} y_1 & y_2 & . & . & . & y_t \\ y_2 & y_3 & . & . & . & y_{t+1} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ y_t & y_{t+1} & . & . & . & y_{2t-1} \end{bmatrix} \begin{bmatrix} \Lambda_t \\ \Lambda_{t-1} \\ . \\ . \\ . \\ \Lambda_1 \end{bmatrix}. \tag{4.16}$$

The coefficients of the error-locator polynomial i.e., $\Lambda$'s can be found by inverting the Toeplitz matrix and multiplying it with the sample measurements. After which the roots of the polynomial will give the error locations $P_l$. Forney's algorithm discussed in Chapter 8 helps to find the values of the errors at these locations. In the next chapter, we shall look into the fundamentals of GSP.

# 5. GRAPH SIGNAL PROCESSING

## 5.1 Introduction

### 5.1.1 From time series signals to graph signals

In the signal processing community, most of the applications use a finite discrete time series signal as shown in Fig. 5.1. A lot of theory has been developed for analyzing discrete-time signals when they are passed through a linear time(shift) invariant system. This idea of shift invariance turns out to be very important in analyzing discrete time signals. In fact, many classical signal processing concepts are built on this idea. Let us now think about shift invariance using a graphical representation as shown in Fig. 5.2, where each node corresponds to a time instant and all the edges are connected with a unit weight based on the fact that each time instant leads to the next time instant. In the Fig. 5.2, node 6 is connected to node 1 assuming that the discrete time signal is a periodic signal. So, we observe that the finite periodic discrete time signals are nothing but signals indexed on a directed cyclic graph. Most of the real world data nowadays like the data from sensor networks, social networks, transportation networks, biological networks etc., can be indexed similarly on a graph where the nodes may be sensors, persons, highways or genes etc., and signals may be temperature, tweets etc. Fig. 5.3 shows one such example of a social networks graph taken from [9] where the nodes represent persons and are connected on the basis of their likings or friendship. It is important to note that this graph has a very irregular structure and is more complex than the one shown in Fig. 5.2.

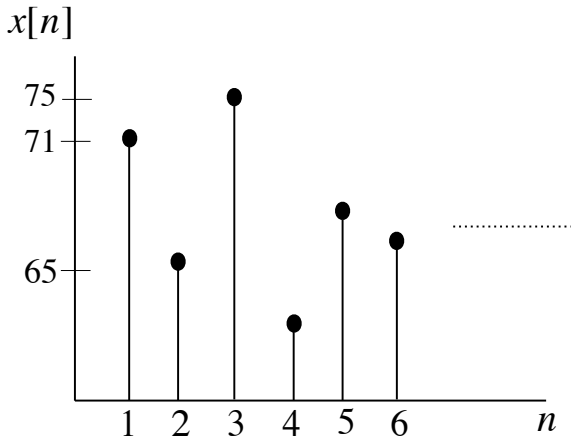



Figure 5.1: Discrete time signal

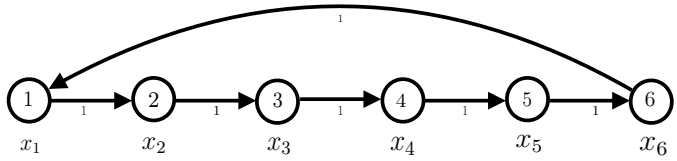



Figure 5.2: Discrete time signal represented in the form of a graph where each node corresponds to a time instant and all the edges are directed with a unit weight

### 5.1.2 Motivation and Literature Survey

Many threads led to the development of Graph Signal Processing(GSP). A classical work in spectral graph theory by [10] links the spectrum of the algebraic representation of graphs to the structure of the graph. Work in image processing by [11] performs image segmentation by considering image in the form of weighted graphs to connect pixels with edges being a function of pixel distance and intensity difference. Work in semi-supervised

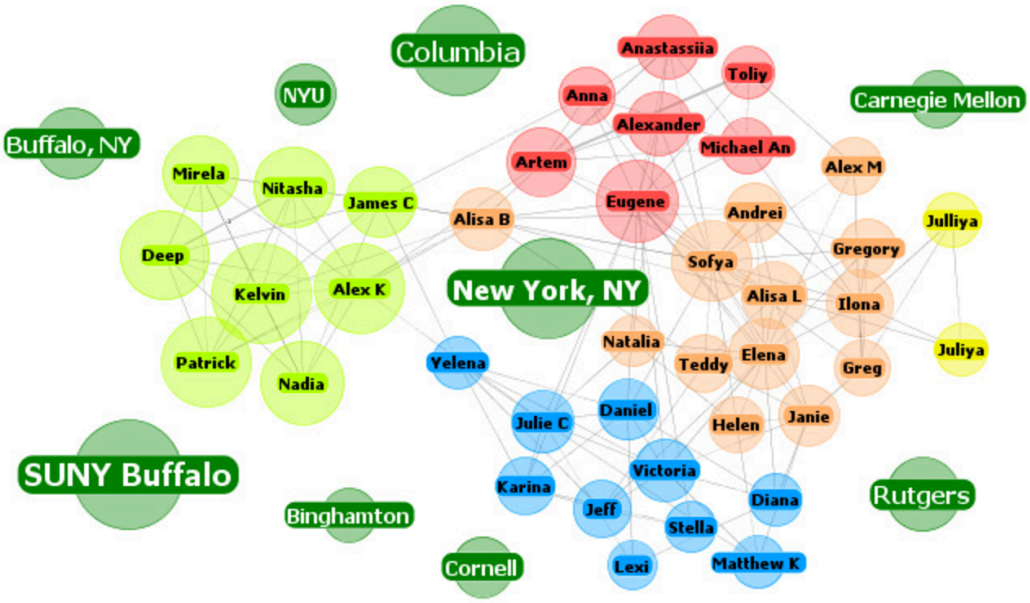



Figure 5.3: A social networks graph where the nodes represent persons and are connected on the basis of their likings or similarity

learning by [12] uses Laplacian regularization on graphs for estimating labels for unlabelled data. Work in network analysis by [13] makes use of the graph transform in vertex domain for the first time to filter information in a node as a function of surrounding nodes. All of these works motivated the development of a generalized framework for signal processing by [14]. Classical signal processing is a specific example of this but in the time domain. Similar things can be performed in space or graph domain[1]. Works by [15] and [16] give a good background on the GSP framework and their applications. It is shown that analogous to DSP, in GSP various techniques such as shift, convolution, Fourier transform, sampling, compressed sensing etc., can be performed.

### 5.1.3 Basic Concepts

#### *5.1.3.1 Graph and Graph Signal*

A graph is a versatile tool to represent the high dimensional and complicated data where the data elements form a node and are connected to each other based on some properties like similarity or physical proximity. Consider a graph represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consisting of $N$ nodes where $\mathcal{V}$ represents a set of nodes $[v_1,v_2,....,v_N]$ and $\mathcal{E}$ represents the edges which are weighted based on some relation/similarity. Edges can be directed or undirected depending upon the applications. For example, in a temperature sensor network each city can be a node and its temperature can be a signal and the cities are connected with edges weighted by the physical distance between them. $\mathcal{E}$ can be represented in matrix form. The two most common matrices used are the adjacency matrix **A** and the Laplacian matrix **L**. These are commonly called as shift operators. The adjacency matrix is an $N \times N$ matrix defined as,

$$A(i,j) = \begin{cases} w_{ij} & (i,j) \in E \geq 0 \\ 0 & otherwise. \end{cases} \tag{5.1}$$

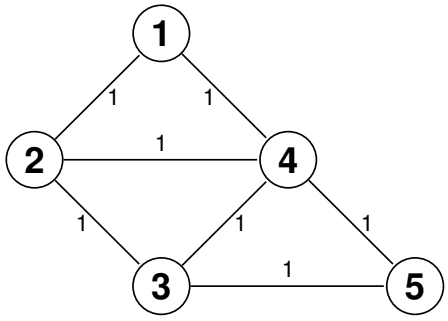
For a general weighted graph, $w_{i,j}$ can be any non-negative real number calculated based on a given similarity function. The Laplacian matrix is defined as,

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{5.2}$$

where, **D** is a diagonal matrix with the diagonal entries being the degree of the corresponding nodes and all other entries being zero. In some cases the normalized versions of **A** and **L** are used. In this thesis, only non-normalized versions are considered.

**Example 1.** *Fig. 5.4 shows an example of a graph with five nodes and the corresponding*

***A** and **L**. It is important to note that these matrices are defined for a fixed ordering of nodes.*



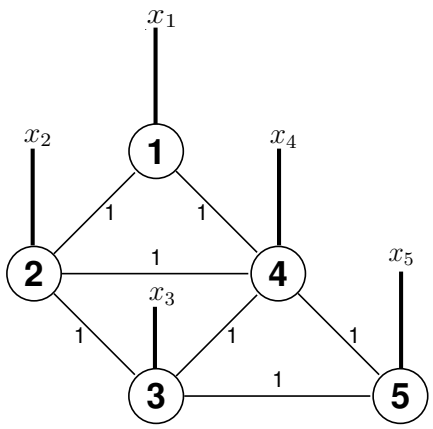

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad L = \begin{bmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$

Figure 5.4: Graph with its **A** and **L**

A graph signal is an attribute associated with the node. Mathematically, a graph signal $x$=$[x_1, x_2, ..., x_N]^T$ is a scalar function defined on vertices of a graph. It can have real or complex values.

**Example 2.** *Fig. 5.5 shows an example of a graph with unity weights, five nodes and a graph signal.*

It can also be noted that the notion of adjacency on a graph tells us about certain similarities of the signals defined on the graph. Classical DSP can be a special case of GSP in the sense, signals defined on a ring graph are nothing but periodic time signals as




$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

Figure 5.5: Graph with signal indexed on each node

already shown in Fig. 5.2. The adjacency matrix in this case is,

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

#### *5.1.3.2 Graph Shift*

In the classical DSP, a shift in the time domain by one involves going from the current time value to either the next time value or the previous time value. But in the graph domain it is highly likely that a node has more than one neighbor. Therefore, a generalized definition of the shift for graphs will be a linear combination of points or graph signals

values around that node weighted by the edge weights. So, a graph shift is defined as,

$$\bar{x} = \mathbf{S}x \tag{5.3}$$

where, **S** is a shift operator which is either the adjacency matrix or the Laplacian matrix. It is interesting to note that for a circulant graph, shifting of the graph signal by one is equivalent to a delay of one in the time domain.

#### *5.1.3.3 Graph Fourier Transform*

The Graph Fourier Transform(GFT) of a graph signal $x$ is the decomposition of the graph signal with respect to an orthonormal eigenvector basis **U** and is given by $\hat{x} = \mathbf{U}^{-1}x$. The inverse GFT is given by $x = \mathbf{U}\hat{x}$. The eigenvector basis **U** can be obtained from either of the two approaches mentioned below and each element of an eigenvector is associated with the corresponding node in the graph.

#### *5.1.3.4 GSP Approaches*

There are two basic GSP approaches depending on the type of operator,

1. The first approach is called the graph spectral approach which is based on the graph Laplacian matrix i.e., $\mathbf{S} = \mathbf{L}$ . It emerged from the spectral graph theory, which is a tool to define frequency spectra for graph Fourier transforms[10]. It is the discrete version of the Laplace operator used in calculus. The graph Laplacian is a real symmetric matrix and for undirected graphs, it is a positive semidefinite matrix too. It has a complete set of orthonormal eigenvectors denoted as **U**=$[u_1, u_2, ..., u_N]$. These eigenvectors have corresponding eigenvalues denoted as **Λ**=$[\lambda_1, \lambda_2, ...\lambda_N]$ satisfying the equation $\mathbf{L}u_l$=$\lambda_l u_l$,$\forall\, l = 1, , ..., N$. So,

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}. \tag{5.4}$$

The graph Fourier basis corresponds to the eigenvectors of the Laplacian matrix. The eigenvalues are called as the graph frequencies. The eigenvectors with large eigenvalues have more sign changes than the ones with small eigenvalues.The notion of bandwidth of a graph signal is given by the maximum eigenvalue of the non-zero frequency coefficients. Since the Laplacian is symmetric and positive semi-definite, the graph frequencies i.e., the eigenvalues are real-valued, nonnegative and ordered. Due to this, Laplacian-based methods are only applicable to undirected graphs with real and non-negative weights. However, [17] suggests a way to find Laplacian for directed graphs.

2. The second approach is called the linear algebra approach which is based on the graph adjacency matrix i.e., $\mathbf{S} = \mathbf{A}$. It emerged from algebraic signal processing theory[14]. The graph Fourier basis corresponds to the eigenvectors of the adjacency matrix which are found by the eigenvalue decomposition of $\mathbf{A}$ given by,

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}. \tag{5.5}$$

However, the frequencies(eigenvalues) are not ordered in this case. Ordering is defined on the basis of a parameter called Total Variation(TV) which is a measure of the similarity between a graph signal and it's shifted version[18] given by,

$$TV(x) = ||x - \mathbf{A}x||_1. \tag{5.6}$$

The frequency components with less variation are called low frequencies and those with more variations are called high frequencies. The notion of bandwidth of a graph signal is given by the number of non-zero frequency coefficients. Adjacency-based methods are applicable to both undirected and directed graphs.

**Remark 1.** *The interesting thing to note is that the choice of basis i.e.,* $\boldsymbol{L}$ *or* $\boldsymbol{A}$ *eigenvectors are graph dependent and not graph signal dependent.*

## 5.2 Sampling of Graph Signals

### 5.2.1 Introduction

Sampling followed by interpolation is one of the most important techniques in classical signal processing. The results from time series signals can be extended to signals defined on graphs. There are two implementation approaches for selecting the nodes namely deterministic and random. In the deterministic approach, a node is either chosen or discarded where as in the random approach the nodes are sampled according to a particular distribution. The former is accurate while the latter is fast and scalable. Typically, there are two sampling strategies commonly used namely experimentally designed sampling and active sampling. While the former is designed based on graph structure(i.e., degree or closeness) only, the later is designed based on the the graph structure and a feedback. The next question that arises is, how can the nodes be sampled. There are three ways available, namely sub-sampling, local sampling and aggregation sampling. Sub-sampling or commonly known as decimation or vertex-based sampling is the same as that of the one used in classical DSP where signals supported on individual nodes are sampled. In local sampling, a linear combination of signals supported on a subset of nodes are sampled. In aggregation sampling, signals supported on individual nodes over time are sampled. This will be discussed in the next section. While sub-sampling and local sampling focus on where to sample in the vertex domain, aggregation sampling trades off between where to sample in the vertex domain and when to sample in the time domain. Before sampling, it is very important to figure out the kinds of signals to be sampled. There are basically three kinds of signals of interest. They are spectrum-aware signals, spectrum-blind signals and vertex-sparse signals. Spectrum-aware signals are the ones whose spectrum(support)

is known and sparse. It is the counter part of the classical bandlimited signal. Spectrum-blind signals are the ones whose spectrum is sparse but the support is unknown. Vertex sparse signals are the ones that are sparse in the graph domain with a certain sparsity. The obvious next step is to apply the various sampling strategies to these signals using either a deterministic or a random approach.

### 5.2.2 Survey of the recent works

One of the first works on sampling of graph signals is reported in [19] which gives the optimality conditions for combinatorial graphs. Work by [20] defines a space signal model where the graph shift is a linear combination of points in a lattice and not the one that goes from left to right (like delay in time domain). The sampling theorem for space models is also stated from continuous space domain to discrete space domain, which completely mimics the classical sampling theorem. Many approaches and strategies are proposed for sampling. [21] uses an experimentally designed sub-sampling method deterministically for signals whose support is known. The key idea is that, the eigenvalue decomposition of the Laplacian is avoided by estimating bandwidth in vertex domain with help of graph spectral proxies after which the sampling set is selected. [22] also works along the same line by applying the adjacency matrix instead. [23] and [24] use an experimentally designed sub-sampling method randomly for signals whose support is known. In [23], the adjacency matrix approach is used and node is sampled based on a sampling score. The key result is that, this is robust to noise. [25] uses an active sub-sampling method randomly for signals whose support is known. This method doesn't quite outperform the previous methods, but empirically it is extremely helpful for smaller graphs due to its feedback mechanism. [26] performs an experimentally designed sub-sampling method randomly for signals whose support is unknown. An $l$-1 optimization framework is used and it outperforms the deterministic approach for the case of irregular graphs.[27] performs a local

sampling for signals whose support is known. Here the linear combination of samples is taken followed by partitioning after which samples from each partition are taken for further sub-sampling. The key result is that, an iterative recovery algorithm is developed for both noiseless and noisy case in which the local averaging weights(related to the bandwidth) are designed according to noise to control the error. [28] uses an active local sampling approach for signals that are sparse in the vertex domain. Due to the feedback mechanism, this can recover supports under small SNR when activated nodes are clustered together. [29] performs an aggregation sampling for signals when the support is known as well as unknown. When the support is known, the key result is that when one node is sampled each time(aggregated samples), perfect recovery is possible under some constraints on the graph. When the support is unknown, sampling strategy is proposed and support identification is done by solving the *l*-1 optimization problem which reduces to the basis pursuit problem.

### 5.2.3 Aggregation Sampling

#### *5.2.3.1 Definition and illustration*

One of the sampling strategies is the aggregation sampling (also known as dynamic sampling) of signals at a single node. Let $\mathcal{G}$ be a graph as discussed above which is provided with a graph shift operator **S** defined as an $N \times N$ matrix whose entry $(i, j)$,denoted as $S_{ij}$, can be non-zero only if $i = j$ or $(j, i) \in \mathcal{E}$. Commonly the shift operator is either the adjacency matrix or the Laplacian. As already mentioned, intuitively the shift gives a linear transformation that can be computed locally at the nodes of the graph. Consider the shifted signal as $y$=$[y_1, y_2, ..., y_N]^T$, then shift is defined as,

$$y = \mathbf{S}x. \tag{5.7}$$

So, node $i$ can compute $y_i$ provided it has access to the values of $x_j$ at its incoming neighbors $j \in \mathcal{V}$. It is assumed that $\mathbf{S}$ is diagonalizable, so there exists an $N \times N$ matrix $\mathbf{U}$ and an $N \times N$ diagonal matrix $\mathbf{\Lambda}$ such that,

$$\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}. \tag{5.8}$$

Let the $l$-th shifted signal be defined as,

$$y^{(l)} = \mathbf{S}^l x \tag{5.9}$$

and the $N \times N$ matrix $\mathbf{Y}$ be defined as,

$$\mathbf{Y} = [y^{(0)}, y^{(1)}, .....y^{(N-1)}] = [x, \mathbf{S}x, ....., \mathbf{S}^{N-1}x]. \tag{5.10}$$

The successively aggregated signal at a selected node $i$ is defined as,

$$y_i = (e_i^T \mathbf{Y})^T = \mathbf{Y}^T e_i \tag{5.11}$$

where, $e_i$ is a $N \times 1$ with all entries being zero except the $i$-th one. We also define a matrix $E_k$=$[e_1, e_2, ..., e_k]$. Equation (5.11) can be written as,

$$y_i = \mathbf{Y}^T e_i = (\mathbf{U}\mathbf{U}^{-1}\mathbf{Y})^T e_i = (\mathbf{U}^{-1}\mathbf{Y})^T \mathbf{U}^T e_i = (\mathbf{U}^{-1}\mathbf{Y})^T v_i \tag{5.12}$$

where, $v_i = \mathbf{U}^T e_i$. Let $\mathbf{C}$ be the $k \times N$ selection matrix which when applied on the signal $x$ choses $k$ out of $N$ elements in $x$. So, the sampled successively aggregated signal at node $i$ is given by,

$$\tilde{y_i} = \mathbf{C}y_i = \mathbf{C}(\mathbf{U}^{-1}\mathbf{Y})^T v_i. \tag{5.13}$$

Fig. 5.6 shows a concise block diagram of this sampling strategy.

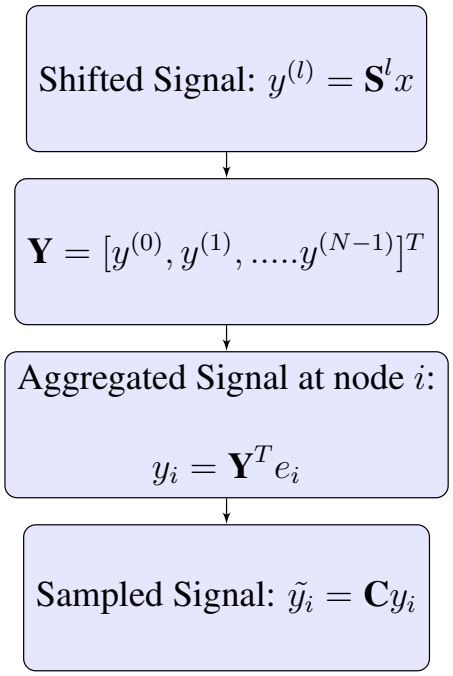



Figure 5.6: Aggregation sampling flow chart

**Example 3.** *A small illustration of this strategy is given here. Consider the undirected graph of* $N = 5$ *nodes as shown in Fig. 5.7. The Shift operator here is the adjacency matrix which is given by,*

$$\boldsymbol{S} = \boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

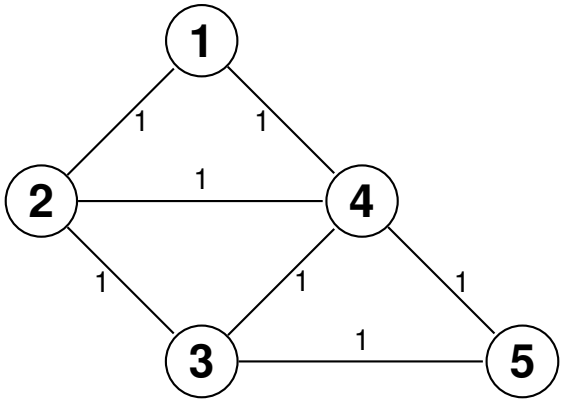


Figure 5.7: Undirected Graph

*After eigenvalue decomposition, we get* $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}$ *where,*

$$\boldsymbol{U} = \begin{bmatrix} 0.3717 & 0.4294 & 0.4390 & 0.6015 & -0.3505 \\ -0.6015 & 0.1378 & -0.5100 & 0.3717 & -0.4700 \\ 0.6015 & 0.1378 & -0.5100 & -0.3717 & -0.4700 \\ 0.0000 & -0.7702 & 0.3069 & -0.0000 & -0.5590 \\ -0.3717 & 0.4294 & 0.4390 & -0.6015 & -0.3505 \end{bmatrix}$$

$$\boldsymbol{\Lambda} = \begin{bmatrix} -1.6180 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -1.4728 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & -0.4626 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.6180 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 2.9354 \end{bmatrix}.$$

*Let* $\boldsymbol{E}$ *be a* $N \times k$ *matrix and* $e_i$ *be its* $i$*-th column.*

$$\boldsymbol{E} = \begin{bmatrix} 1 & 0 & 0 & . & . & . \\ 0 & 1 & 0 & . & . & . \\ 0 & 0 & 1 & . & . & . \\ 0 & 0 & 0 & . & . & . \\ 0 & 0 & 0 & . & . & . \end{bmatrix}.$$

*Let the selection matrix be* $\boldsymbol{C} = \boldsymbol{E}_k^T$*. We assume that we know the positions of the non-zero frequency coefficients. Let it be the first two coefficients i.e., sparsity is* $k$ *=2. Let the graph domain signal be,*

$$x = \begin{bmatrix} 0.203 \\ -0.0789 \\ 0.1617 \\ -0.2311 \\ 0.0545 \end{bmatrix}.$$

*So, the frequency domain signal is* $\hat{x} = \boldsymbol{U}^{-1}x$ *given by,*

$$\hat{x} = \begin{bmatrix} 0.2 \\ 0.3 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

*Let* $\boldsymbol{Y} = [x, \boldsymbol{A}x, \boldsymbol{A}^2x, \boldsymbol{A}^3x, \boldsymbol{A}^4x]$ *contain the shifted versions of the signal, so*

$$\boldsymbol{Y} = \begin{bmatrix} x_1 & x_2+x_4 & 2x_1+x_2+x_4+x_5 & 2x_1+5x_2+3x_3+6x_4+3x_5 & 11x_1+11x_2+14x_3+13x_4+8x_5 \\ x_2 & x_1+x_3+x_4 & x_1+3x_2+x_3+2x_4+2x_5 & 5x_1+4x_2+7x_3+7x_4+3x_5 & 11x_1+19x_2+14x_3+17x_4+12x_5 \\ x_3 & x_2+x_4+x_5 & 2x_1+x_2+3x_3+2x_4+x_5 & 3x_1+7x_2+4x_3+5x_4+4x_5 & 14x_1+14x_2+19x_3+19x_4+10x_5 \\ x_4 & x_1+x_2+x_3+x_4+x_5 & x_1+2x_2+2x_3+4x_4+x_5 & 6x_1+7x_2+7x_3+6x_4+5x_5 & 13x_1+19x_2+19x_3+24x_4+12x_5 \\ x_5 & x_3+x_4 & x_1+2x_2+x_3+x_4+x_5 & 3x_1+3x_2+5x_3+6x_4+2x_5 & 9x_1+10x_2+12x_3+12x_4+7x_5 \end{bmatrix}.$$

*We assume the node selected is* $i = 3$*. The aggregated signal at node* $3$ *is given by,*

$$y_3 = \boldsymbol{Y}^T e_3 = \begin{bmatrix} x_3 \\ x_2 + x_4 + x_5 \\ 2x_1 + x_2 + 3x_3 + 2x_4 + x_5 \\ 3x_1 + 7x_2 + 4x_3 + 5x_4 + 4x_5 \\ 14x_1 + 14x_2 + 19x_3 + 19x_4 + 10x_5 \end{bmatrix} = \begin{bmatrix} 0.1617 \\ -0.2556 \\ 0.4047 \\ -0.6417 \\ 1.0191 \end{bmatrix}.$$

*So, the sampled signal is,*

$$\tilde{y_3} = \boldsymbol{C} y_3 = \boldsymbol{E}_2^T y_3 = \begin{bmatrix} 0.1617 \\ -0.2556 \end{bmatrix}. \tag{5.14}$$

*Fig. 5.8 shows an intuitive view of this sampling strategy (only the first two shifts are shown).*

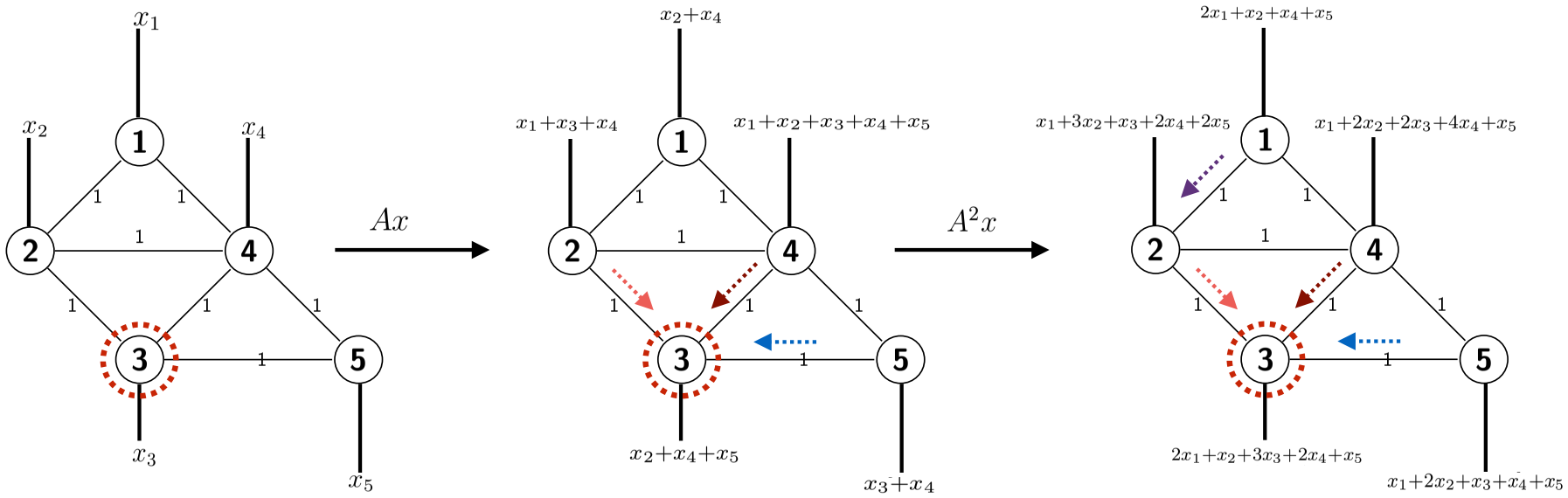



Figure 5.8: Intuition of aggregation sampling

# 6. SPECTRAL ESTIMATION FOR GRAPH SIGNALS USING RS DECODING

In the previous chapter, we discussed a method to sample a graph signal. In this chapter, we propose an efficient reconstruction algorithm to recover the $k$-sparse graph signal, given the $2k$ samples obtained using the aggregation sampling strategy. We exploit the Vandermonde structure in the sampling strategy to build a less complex algorithm, which is discussed in the following section.

## 6.1 Presence of Vandermonde Structure

From equation (5.10) the $l$-th column of $\mathbf{Y}$ is given by,

$$\mathbf{S}^{l-1}x = \mathbf{U}\mathbf{\Lambda}^{l-1}\mathbf{U}^{-1}x = \mathbf{U}\mathbf{\Lambda}^{l-1}\hat{x}. \tag{6.1}$$

Therefore, the $l$-th column of $\mathbf{U}^{-1}\mathbf{Y}$ is given by,

$$\mathbf{U}^{-1}\mathbf{U}\mathbf{\Lambda}^{l-1}\hat{x} = \mathbf{\Lambda}^{l-1}\hat{x} = \text{diag}(\hat{x})[\lambda_1^{l-1}, \lambda_2^{l-1}, ....\lambda_N^{l-1}]^T. \tag{6.2}$$

Let the powers of the $\lambda$'s for $l = 1, 2..., N$ be stored in a matrix which is defined as,

$$\mathbf{\Psi} = \begin{bmatrix} 1 & 1 & . & . & . & 1 \\ \lambda_1 & \lambda_2 & . & . & . & \lambda_N \\ \lambda_1^2 & \lambda_2^2 & . & . & . & \lambda_N^2 \\ \lambda_1^3 & \lambda_2^3 & . & . & . & \lambda_N^3 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \lambda_1^{N-1} & \lambda_2^{N-1} & . & . & . & \lambda_N^{N-1} \end{bmatrix}.$$

We observe that this matrix turns out to be a Vandermonde matrix. So, $\mathbf{U}^{-1}\mathbf{Y}$ can be written as,

$$\mathbf{U}^{-1}\mathbf{Y} = \text{diag}(\hat{s})\mathbf{\Psi}^T. \tag{6.3}$$

Now, substituting (6.3) in (5.12)

$$y_i = (\mathbf{U}^{-1}\mathbf{Y})^T v_i = [\text{diag}(\hat{x})\mathbf{\Psi}^T]^T v_i = \mathbf{\Psi}\text{diag}(\hat{x})v_i = \mathbf{\Psi}\text{diag}(v_i)\hat{x}. \tag{6.4}$$

Let $\mathbf{C}$ be the $k \times N$ selection matrix which when applied on the signal $x$ choses $k$ out of $N$ elements in $x$. The sampled successively aggregated signal at node $i$ is given by,

$$\tilde{y_i} = \mathbf{C}y_i = \mathbf{C}\mathbf{\Psi}\text{diag}(v_i)\hat{x}. \tag{6.5}$$

Using the same example as in the previous section,

$$v_3 = \mathbf{U}^T e_3 = \begin{bmatrix} 0.6015 \\ 0.1378 \\ -0.5100 \\ -0.3717 \\ -0.4700 \end{bmatrix}.$$

The Vandermonde matrix can be formed as,

$$\mathbf{\Psi} = \begin{bmatrix} 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 \\ -1.6180 & -1.4728 & -0.4626 & 0.6180 & 2.9354 \\ 2.6180 & 2.1692 & 0.2140 & 0.3820 & 8.6168 \\ -4.2361 & -3.1949 & -0.0990 & 0.2361 & 25.2939 \\ 6.8541 & 4.7056 & 0.0458 & 0.1459 & 74.2486 \end{bmatrix}.$$

So,

$$y_3 = \mathbf{\Psi}\text{diag}(v_3)\hat{x} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 & \lambda_5 \\ \lambda_1^2 & \lambda_2^2 & \lambda_3^2 & \lambda_4^2 & \lambda_5^2 \\ \lambda_1^3 & \lambda_2^3 & \lambda_3^3 & \lambda_4^3 & \lambda_5^3 \\ \lambda_1^4 & \lambda_2^4 & \lambda_3^4 & \lambda_4^4 & \lambda_5^4 \end{bmatrix} \begin{bmatrix} v_{11} & 0 & 0 & 0 & 0 \\ 0 & v_{22} & 0 & 0 & 0 \\ 0 & 0 & v_{33} & 0 & 0 \\ 0 & 0 & 0 & v_{44} & 0 \\ 0 & 0 & 0 & 0 & v_{55} \end{bmatrix} \begin{bmatrix} \hat{x_1} \\ \hat{x_2} \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (6.6)$$

Therefore,

$$y_3 = \begin{bmatrix} v_{11}\hat{s_1} + v_{22}\hat{x_2} \\ \lambda_1 v_{11}\hat{x_1} + \lambda_2 v_{22}\hat{x_2} \\ \lambda_1^2 v_{11}\hat{x_1} + \lambda_2^2 v_{22}\hat{x_2} \\ \lambda_1^3 v_{11}\hat{x_1} + \lambda_2^3 v_{22}\hat{x_2} \\ \lambda_1^4 v_{11}\hat{x_1} + \lambda_2^4 v_{22}\hat{x_2} \end{bmatrix} = \begin{bmatrix} 0.1617 \\ -0.2556 \\ 0.4047 \\ -0.6417 \\ 1.0191 \end{bmatrix}.$$

The sampled version of $y_3$ is given by,

$$\tilde{y_3} = \mathbf{C}y_3 = \mathbf{E}_2^T y_3 = \begin{bmatrix} v_{11}\hat{x_1} + v_{22}\hat{x_2} \\ \lambda_1 v_{11}\hat{s_1} + \lambda_2 v_{22}\hat{x_2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.1617 \\ -0.2556 \\ 0.4047 \\ -0.6417 \\ 1.0191 \end{bmatrix}.$$

Hence,

$$\tilde{y_3} = \begin{bmatrix} 0.1617 \\ -0.2556 \end{bmatrix}. \quad (6.7)$$

Equations (5.14) and (6.7) are the same which shows the equivalency in both domains. The main take way from this derivation is equation (6.5) which gives a way for the proposed

work.

## 6.2 Proposed algorithm

Consider equation (6.5),

$$\tilde{y_i} = \mathbf{C}y_i = \mathbf{C}\mathbf{\Psi}\text{diag}(v_i)\hat{x}. \tag{6.8}$$

Since $\mathbf{\Psi}$ is a Vandermonde matrix, $\mathbf{C}\mathbf{\Psi}$ also has a Vandermonde structure. As $\hat{x}$ is $k$-sparse that means the term $\text{diag}(v_i)\hat{x}$ is also a $k$-sparse vector. Let $\mathbf{C}\mathbf{\Psi} = \mathbf{H}$ and $\text{diag}(v_i)\hat{x} = e$ and $\tilde{y_i} = y'$. Then equation (6.8) can be written as,

$$y' = \mathbf{H}e. \tag{6.9}$$

Now, this equation has a structure which is same as the ones we have seen before in equations (2.16) and (3.3) i.e., the syndrome and sensing matrix equations respectively. This means that taking ideas from Chapter 2.2.2 we can derive an equivalent RS decoding for GSP. Lets derive in the form of an example and then generalize it.

**Example 4.** *Consider the same graph as in Example 1 and 2. Equation (6.6) can be written as,*

$$y' = \boldsymbol{C}y_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 & \lambda_5 \\ \lambda_1^2 & \lambda_2^2 & \lambda_3^2 & \lambda_4^2 & \lambda_5^2 \\ \lambda_1^3 & \lambda_2^3 & \lambda_3^3 & \lambda_4^3 & \lambda_5^3 \end{bmatrix} \begin{bmatrix} v_{11}\hat{x_1} \\ v_{22}\hat{x_2} \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

*Let $e_1 = v_{11}\hat{x_1}$ and $e_2 = v_{22}\hat{x_2}$. This equivalently can be written as,*

$$\begin{bmatrix} y_1' \\ y_2' \\ y_3' \\ y_4' \end{bmatrix} = \begin{bmatrix} e_1 + e_2 \\ e_1\lambda_1 + e_2\lambda_2 \\ e_1\lambda_1^2 + e_2\lambda_2^2 \\ e_1\lambda_1^3 + e_2\lambda_2^3 \end{bmatrix}.$$

*Let us form a polynomial similar to the polynomial in equations (4.6) and (4.14),*

$$\Gamma(w) = (1 - \lambda_1 w)(1 - \lambda_2 w) = 1 + \Gamma_1 w + \Gamma_2 w^2. \tag{6.10}$$

*The roots of this polynomial are $\lambda_1^{-1}$ and $\lambda_2^{-1}$. So, $\Gamma(\lambda_1^{-1}) = 0$ and $\Gamma(\lambda_2^{-1}) = 0$. Therefore,*

$$\Gamma(\lambda_1^{-1}) = \Gamma_2\lambda_1^{-2} + \Gamma_1\lambda_1^{-1} + 1 = 0 \tag{6.11}$$

*and*

$$\Gamma(\lambda_2^{-1}) = \Gamma_2\lambda_2^{-2} + \Gamma_1\lambda_2^{-1} + 1 = 0. \tag{6.12}$$

*Multiplying equation (6.11) by $e_1\lambda_1^2$ and (6.12) by $e_2\lambda_2^2$,*

$$e_1\lambda_1^2\Gamma(\lambda_1^{-1}) = \Gamma_2 e_1 + \Gamma_1 e_1\lambda_1 + e_1\lambda_1^2 = 0 \tag{6.13}$$

*and*

$$e_2\lambda_2^2\Gamma(\lambda_2^{-1}) = \Gamma_2 e_2 + \Gamma_1 e_2\lambda_2 + e_2\lambda_2^2 = 0. \tag{6.14}$$

*Adding equations (6.13) and (6.14) we get,*

$$\Gamma_2(e_1 + e_2) + \Gamma_1(e_1\lambda_1 + e_2\lambda_2) + (e_1\lambda_1^2 + e_2\lambda_2^2) = 0. \tag{6.15}$$

*Simplifying this we get,*

$$\Gamma_2 y_1' + \Gamma_1 y_2' + y_3' = 0. \tag{6.16}$$

*Similarly, repeating this procedure again by multiplying equations (6.11) and (6.12) by* $e_1\lambda_1^3$ *and* $e_2\lambda_2^3$ *respectively and proceeding in the same way we get,*

$$\Gamma_2 y_2' + \Gamma_1 y_3' + y_4' = 0. \tag{6.17}$$

*Writing equation (6.9) and (6.10) in the matrix form we get,*

$$\begin{bmatrix} -y_3' \\ -y_4' \end{bmatrix} = \begin{bmatrix} y_1 & y_2 \\ y_2 & y_3 \end{bmatrix} \begin{bmatrix} \Gamma_2 \\ \Gamma_1 \end{bmatrix}. \tag{6.18}$$

*The coefficients of this polynomial can be found by the inverting the* $2 \times 2$ *Toeplitz matrix. The roots of the polynomial gives the required* $\lambda$*'s.*

Generalizing this to any $k$-sparse signal, we get,

$$\begin{bmatrix} -y_{k+1}' \\ -y_{k+2}' \\ . \\ . \\ . \\ -y_{2k}' \end{bmatrix} = \begin{bmatrix} y_1 & y_2 & . & . & . & y_k \\ y_2 & y_3 & . & . & . & y_{k+1} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ y_k & y_{k+1} & . & . & . & y_{2k-1} \end{bmatrix} \begin{bmatrix} \Gamma_k \\ \Gamma_{k-1} \\ . \\ . \\ . \\ \Gamma_1 \end{bmatrix}. \tag{6.19}$$

This equation is similar to equation (4.16) thus proving the justification that the RS decoding is indeed applicable for graph signals. This way of forming the polynomial is also done by the Berlekamp-Massey(BM) algorithm. The roots gives us the locations of the non-zero elements in the $k$-sparse vector. The values of these non-zero elements are com-

puted by the Forney's algorithm. In the next two sections, we shall discuss briefly about the BM and Forney's algorithm.

### 6.2.1 Berlekamp-Massey algorithm for complex field

Berlekamp-Massey algorithm[30] is an algorithm that computes the error locator polynomial by finding the shortest linear feedback register. The inputs to this algorithm are a set of syndromes($S$'s) computed while decoding RS codes. Let $L$ be the number of errors and $N$ be the number of syndromes. The error locator polynomial $P(x)$ for $L$ errors is defined as,

$$P(x) = 1 + P_1 x + P_2 x^2 + ... + P_{L-1} x^{L-1} + P_L x^L. \tag{6.20}$$

The aim of the algorithm is to determine the polynomial $P(x)$ with minimum degree $L$ i.e., the coefficients $P_1, P_2, ...P_L$ which result in all the syndromes such that,

$$S_n + P_1 S_{n-1} + ... + P_L S_{n-L} = 0, \quad L \leq n \leq N-1.$$

$P(x)$ is initialized to 1 and $L$ is initialized to zero. $n$ is the iterator which changes from 0 to $N-1$. $B(x)$ is a copy of the last $P(x)$ and initialized to 1, $d$ is the discrepancy, $b$ is the copy of the last discrepancy $d$ and is initialized to 1, $m$ is the number of iterations and initialized to 1.

Each iteration of the algorithm calculates the discrepancy $d$. At the $k$-th iteration $d$ is given by,

$$d = S_k + P_1 S_{k-1} + ... + P_L S_{k-L} = 0. \tag{6.21}$$

If $d = 0$, the algorithm assumes that $P(x)$ and $L$ are correctly found and proceeds by incrementing $m$. If $d \neq 0$, $P(x)$ is modified as,

$$P(x) = P(x) - \left(\frac{d}{b}\right) x^m B(x). \tag{6.22}$$

The $x^m$ term shifts $B(x)$, so that it follows the syndromes corresponding to $b$. The discrepancy $d$ is again recalculated as,

$$d = S_k + P_1 S_{k-1} + ... - \left(\frac{d}{b}\right)(S_j + B_1 S_{j-1} + ...) = 0. \tag{6.23}$$

After this $L$ is also incremented. This process continues until all the syndromes are covered and finally the output is the error locator polynomial whose roots can be found numerically using the MATLAB command $roots()$.

### 6.2.2 Forney's Algorithm

Once the error locator polynomial and its roots are found, the next step is to find out the error values. The inputs to this algorithm are the syndromes and the error locator polynomial. Consider the syndrome equation given by,

$$S_j = \sum_{l=1}^{L} e_{i_l} X_l^j \qquad \text{for} \quad j = 1, 2..., 2t \tag{6.24}$$

where, $L$ is the number of errors in the received codeword and $t$ is the error correcting ability of the RS code. With the knowledge of the error locators obtained from the roots of the error locator polynomial, we can setup and solve a set of linear equations:

$$\begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ . \\ . \\ S_{2t} \end{bmatrix} = \begin{bmatrix} X_1 & X_2 & . & . & . & X_L \\ X_1^2 & X_2^2 & . & . & . & X_L^2 \\ X_1^3 & X_2^3 & . & . & . & X_L^3 \\ X_1^4 & X_2^4 & . & . & . & X_L^4 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ X_1^{2t} & X_2^{2t} & . & . & . & X_L^{2t} \end{bmatrix} \begin{bmatrix} e_{i_1} \\ e_{i_2} \\ e_{i_3} \\ e_{i_4} \\ . \\ . \\ e_{i_L} \end{bmatrix}. \tag{6.25}$$

Let us define a syndrome polynomial as,

$$S(x) = S_1 + S_2x + S_3x^2 + ... + S_{2t}x^{2t-1} = \sum_{j=0}^{2t-1} S_{j+1}x^j. \tag{6.26}$$

Also, let an error evaluator polynomial be defined as,

$$\omega(x) = S(x)P(x) \quad (\text{mod } x^{2t}) \tag{6.27}$$

where, $P(x)$ is the error locator polynomial with roots $X_l^{-1}$ . The modulo $x^{2t}$ discards all the terms of degree greater than or equal to $2t$. Then the error values for an RS code are given by[30],

$$e_{i_k} = -\frac{\omega(X_k^{-1})}{P'(X_k^{-1})} \tag{6.28}$$

where, $P'(x)$ is the derivative of $P(x)$.

# 7. MULTIPLE ACCESS COMMUNICATION CHANNEL

Consider a set of $N$ sensors distributed in a geographic region as shown in the Fig. 7.1. Each sensor measures some physical quantity like temperature, humidity etc. All the sensors are connected wirelessly to a base station. They transmit the signals(measurements) i.e., $x_i$'s simultaneously through a Multiple Access Communication(MAC) channel to the base station as shown in Fig. 7.1. The signal is considered to be $k$-sparse in the GFT domain. The base station receives a sum of the $x_i$'s. The transmission process happens at the expense of power(energy). The objective is to maximize the power efficiency of the network. In this chapter, we discuss two MAC schemes. The first scheme is the Time Division Multiple Access(TDMA) scheme which serves as a benchmark and the second scheme is the proposed scheme based on GSP.

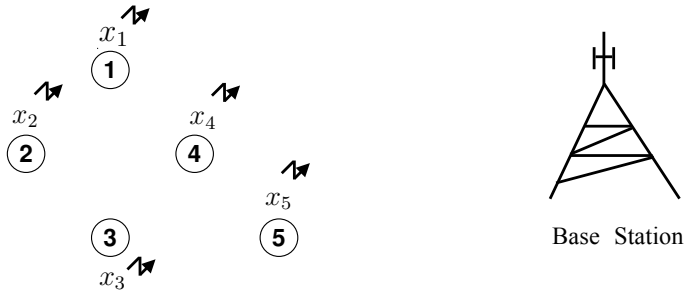



Figure 7.1: MAC

## 7.1 TDMA scheme

A naive solution is to use the Time Division Multiple Access(TDMA) scheme where each sensor is given a time slot to send its measurement. We ignore the dependence in the

measurements between the sensors. There are $N$ time slots(one per sensor) as shown in the Fig. 7.2. Let the total time frame be $N$ seconds long for the $N$ slots, $B$ be the length

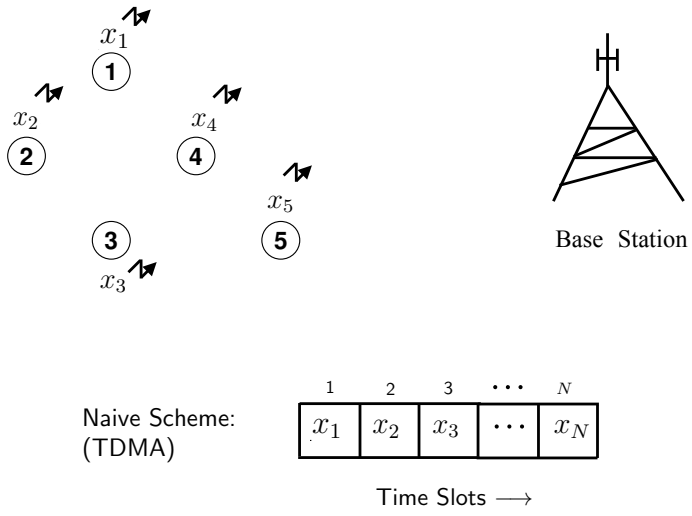



Figure 7.2: TDMA based scheme

of the message(in bits) sent by each sensor, $E_t$ be energy per channel use. One use of a channel happens when a sensor sends all the $B$ bits in one time slot. So, the total energy consumed by all users for $N$ slots is $NE_t$. We know that the capacity of a TDMA scheme per user is,

$$\text{Capacity} = \frac{1}{2}\log\left(1 + \frac{E_t}{\sigma^2}\right). \tag{7.1}$$

In order for the $B$ bits to be transmitted successfully we require that,

$$\frac{1}{2}\log\left(1 + \frac{E_t}{\sigma^2}\right) > B.$$

Simplifying this gives,

$$E_t > 2^{2B}\sigma^2.$$

Therefore, the total energy required is $E_{tot} = NE_t = N2^{2B}\sigma^2$. We observe that total energy varies linearly with $N$.

### 7.2 Proposed scheme

Let us consider the set of $N$ sensors distributed in a geographic region as nodes in a graph and the measurement like temperature to be a graph signal. An example is shown in the Fig. 7.3. We employ the following procedure based on our proposed work,

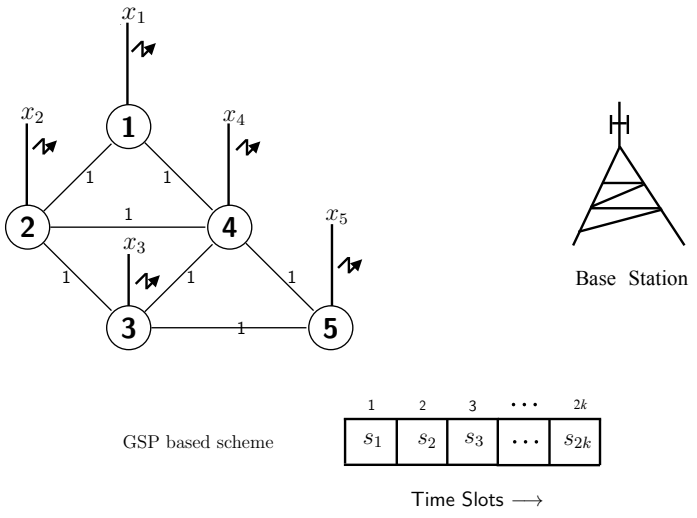



Figure 7.3: GSP based scheme

1. Given the graph structure, the first $2k$ powers of shift operator **S** is computed. In this case, we consider **S** to be the **A** matrix.

2. A suitable node $i$ is chosen which participates in the specific frequencies on which signal $x$ is expressed (i.e $e_k^T v_i \neq 0$).

3. A matrix called the scaled factor matrix is formed by selecting the $i$th row of the

first $2k$ powers of shift operator **A** and is represented as **Z**. Example 5 shows a clear picture of the formation of the scaled factor matrix.

4. The graph signal is scaled using the **Z** matrix. For example, in slot 1 signal $x_1$ is scaled by a factor of $z_{11}$, $x_2$ is scaled by a factor of $z_{12}$ and so on. Each of these scaled versions are sent simultaneously to the base station.

5. The channel adds up these scaled signals forming a linear combination of the signal components. The base station receives these linear combinations and with the knowledge of the scaled factor matrix decodes the signals with the help of the RS decoding scheme for graph signals mentioned in the previous section.

Let the scaled matrix formed be given as,

$$\mathbf{Z} = \begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ z_{2k1} & z_{2k2} & . & . & z_{2kN} \end{bmatrix}.$$

So the scaled signal is given by,

$$s = \begin{bmatrix} s_1 \\ s_2 \\ . \\ . \\ . \\ s_{2k} \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ z_{2k1} & z_{2k2} & . & . & z_{2kN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_N \end{bmatrix} = \begin{bmatrix} z_{11}x_1 + z_{12}x_2 + .... \\ z_{21}x_1 + z_{22}x_2 + ... \\ . \\ . \\ . \\ z_{2k1}x_1 + z_{2k2}x_2 + ... \end{bmatrix}.$$

**Example 5.** *Consider the graph as shown in Fig. 7.3. The first four powers (0 to 2k-1)of* ***A*** *are given by,*

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 2 & 1 & 2 & 1 & 1 \\ 1 & 3 & 1 & 2 & 2 \\ 2 & 1 & 3 & 2 & 1 \\ 1 & 2 & 2 & 4 & 1 \\ 1 & 2 & 1 & 1 & 2 \end{bmatrix}, \begin{bmatrix} 2 & 5 & 3 & 6 & 3 \\ 5 & 4 & 7 & 7 & 3 \\ 3 & 7 & 4 & 7 & 5 \\ 6 & 7 & 7 & 6 & 6 \\ 3 & 3 & 5 & 6 & 2 \end{bmatrix}.$$

*Now, considering that the $3^{rd}$ node is chosen, the $3^{rd}$ row from each of these matrices is picked and the scaled factor matrix is formed as given below,*

$$\boldsymbol{Z} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 2 & 1 & 3 & 2 & 1 \\ 3 & 7 & 4 & 5 & 4 \end{bmatrix}.$$

*Finally, the scaled graph signal is formed as,*

$$s = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 2 & 1 & 3 & 2 & 1 \\ 3 & 7 & 4 & 5 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_2 + x_4 + x_5 \\ 2x_1 + x_2 + 3x_3 + 2x_4 + x_5 \\ 3x_1 + 7x_2 + 4x_3 + 5x_4 + 4x_5 \end{bmatrix}.$$

*The base station receives each of $s_1, s_2, ...s_{2k}$ in a slot. For example, the base station receives $x_3$ in slot 1, $x_2 + x_4 + x_5$ in slot 2 and so on.*

A model of the GSP based scheme is shown in Fig. 7.4. Let the total time frame be same as TDMA i.e., $N$ seconds but for the $2k$ slots. $B$ be the length of the message(in

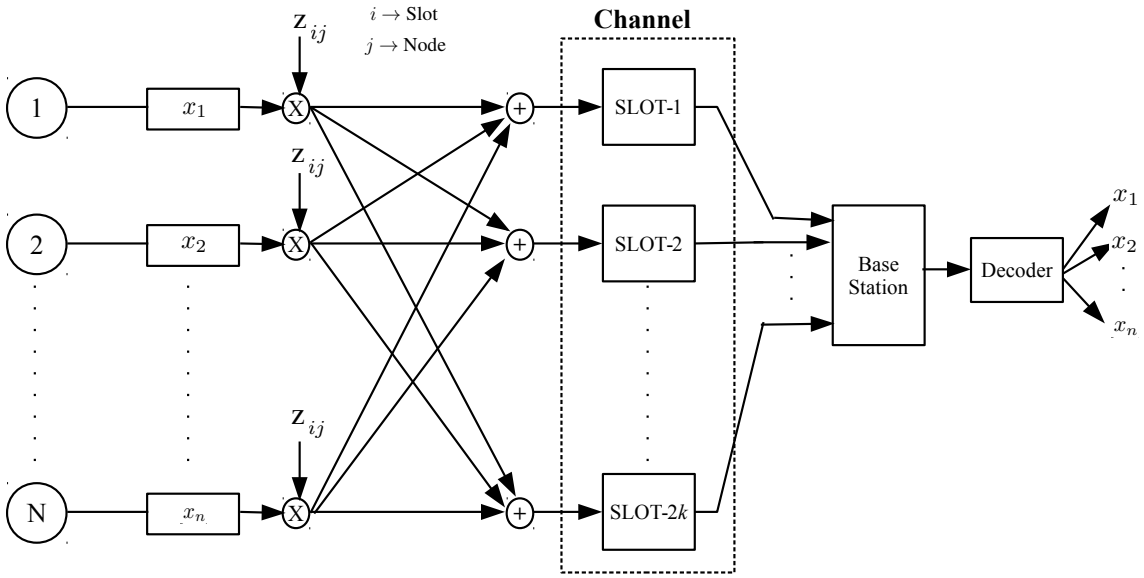



Figure 7.4: Proposed Model

bits) sent by each sensor and $E_g$ be energy per channel use. The number of channel uses in one slot will be $\frac{N}{2k}$. So, the energy consumed per user is $\frac{N}{2k} \times 2kE_s = NE_s$. Therefore, the total energy for $N$ users is $NE_s \times N = N^2E_s$. Assuming we use lattice codes at the physical layer, $s_i$'s can be decoded if,

$$\frac{N}{4k}\log\left(\frac{E_s}{\sigma^2}\right) > B.$$

Simplifying this gives,

$$E_s > 2^{\frac{4kB}{N}}\sigma^2. \tag{7.2}$$

Therefore, the total energy required is,

$$E_{tot} = N^2E_s = N^2 \times 2^{\frac{-N}{4kB}}\sigma^2 = \frac{N^2}{2^{\frac{N}{4kB}}}\sigma^2. \tag{7.3}$$

We observe that energy varies(decreases) exponentially with $N$. Also, the observations i.e., the linear combinations contained in $s_i$'s are computed naturally by the channel model, so no extra cost is incurred here. Therefore, this scheme is more power(energy) efficient and less complex than the TDMA scheme.

# 8. ANOMALY DETECTION

Consider a sensor network with $N$ nodes which are connected on the basis of some similarity. Let the signal indexed on the nodes be $k$-bandlimited i.e., few($k$) low frequency coefficients are active and many high frequency coefficients are zero. Fig. 8.1 shows an example of a graph signal that is bandlimited in the underlying graph. The aim is to identify the malfunctioning nodes(anomalies) and correct the erroneous signal.

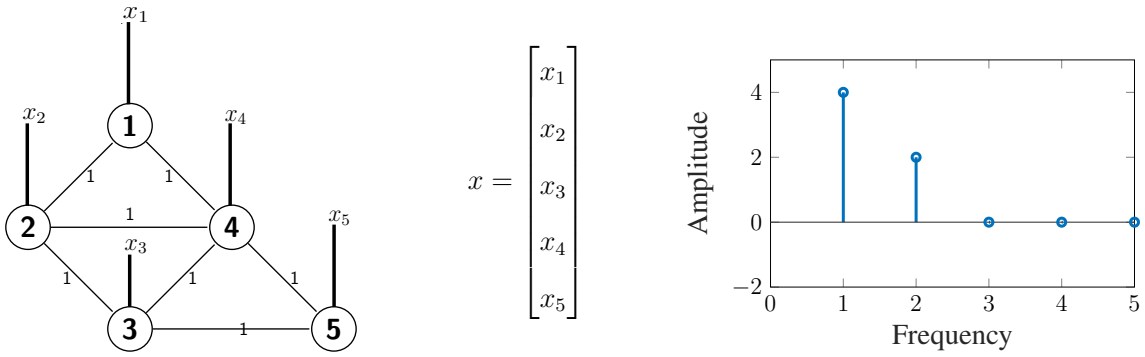



Figure 8.1: Graph with a sparse signal

Let $x$ be the signal defined on the nodes. Since the signal is $k$-bandlimited that means the GFT of $\mathcal{F}(x) = 0$ for frequency coefficients more than $k$. Suppose few of the nodes malfunction so that the signal is perturbed by some error i.e.,

$$x' = x + e. \tag{8.1}$$

So, the high frequency coefficients are no more zero. Fig. 8.2 shows the perturbed signal due to the malfunctioning of nodes 1 and 3.

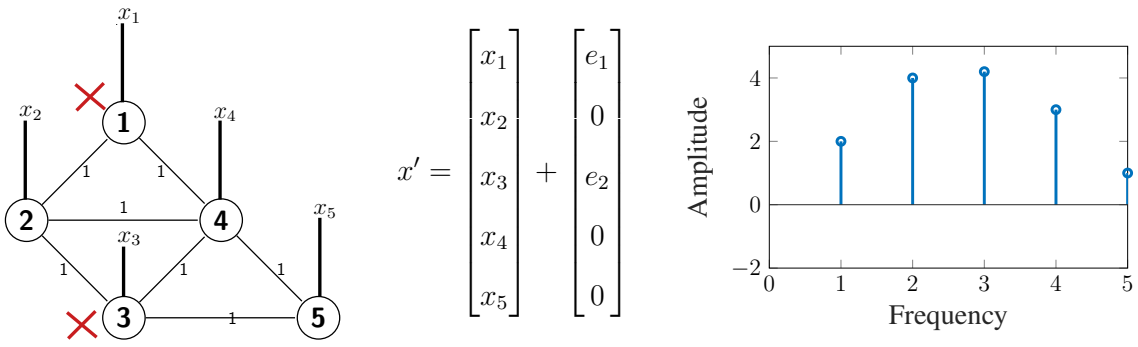



Figure 8.2: Graph with a perturbed signal

The GFT of the perturbed signal will now be given as,

$$\mathcal{F}(x') = \mathcal{F}(x + e) = \mathcal{F}(x) + \mathcal{F}(e). \tag{8.2}$$

If only the high frequency coefficients are considered say from index $k + 1$ to $N$, then

$$\mathcal{F}(x') = \mathcal{F}(x) + \mathcal{F}(e) = 0 + \mathcal{F}(e). \tag{8.3}$$

The aim is to compute the malfunctioning nodes(anomalies) and the locations where they are present. We frame this as a CS problem by considering the $\mathcal{F}(e)$ as the measurements and the $k + 1$ to $N$ columns of the eigenvector matrix as the sensing matrix and apply algorithms like LASSO and Orthogonal Matching Pursuit(OMP) to obtain the sparse error vector and their (malfunctioning node) locations. Fig. 8.3 shows a block diagram of the approach.

For simulation purpose, a real temperature data from sensors located at 150 major US cities is taken. The dataset is available at `ftp://ftp.ncdc.noaa.gov/pub/data/gsod`. The sensors are represented in the form of a graph using geographical distances between them as shown in Fig. 8.4. Here each sensor corresponds to a node and

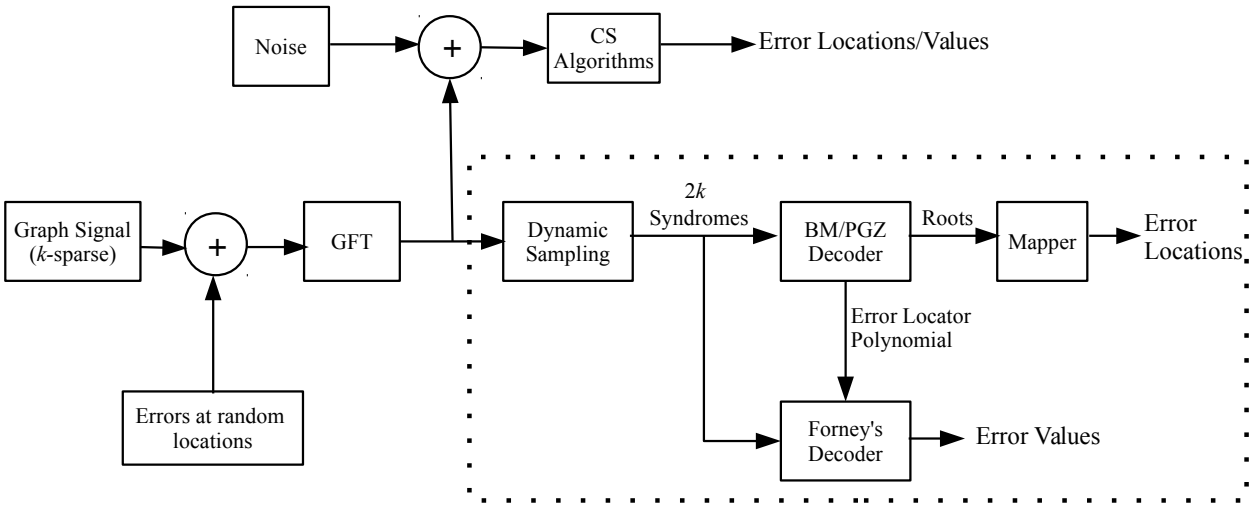



Figure 8.3: Anomaly Detection Model

is connected to its six nearest sensors with undirected edges weighted by a factor given in[18]. The temperature values are considered as graph signal.

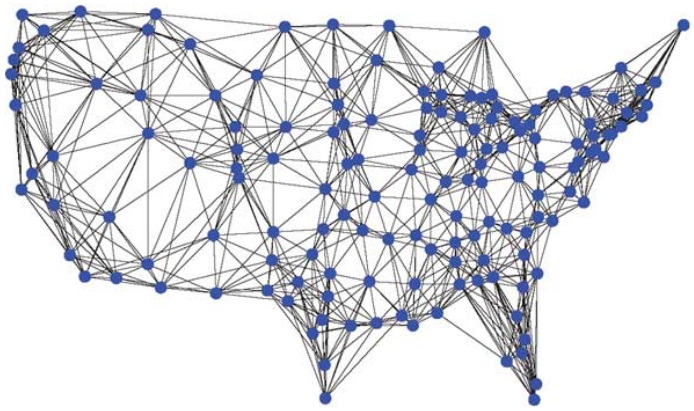


Figure 8.4: Temperature sensor graph formed using the real temperature dataset

The analysis is carried out using both the approaches for GSP i.e., the graph Laplacian

matrix and graph adjacency matrix approach. Anomalies(errors) in the range of [-20,20] were added and the sparsity(number of errors) was changed from 1 to 100. The recovery algorithms LASSO and OMP were implemented for these two approaches. The success rate(fraction of correctly identified errors over the total number of errors) and their values are plotted against the sparsity for all the four cases as shown in Fig. 8.5. Both the recovery algorithms detect the anomalies fairly well with more than 90% accuracy when the number of anomalies are below 10. In particular, the adjacency matrix based approach with OMP performs the best for all the values of anomalies. Since the notion of bandwidth of a graph signal in the adjacency matrix approach is given by the number of non-zero frequency coefficients which is analogous to the classical DSP, so this may be a reason for this approach to do well with CS algorithms. However, this may not be concluded for all the applications i.e., it may be application/graph specific.

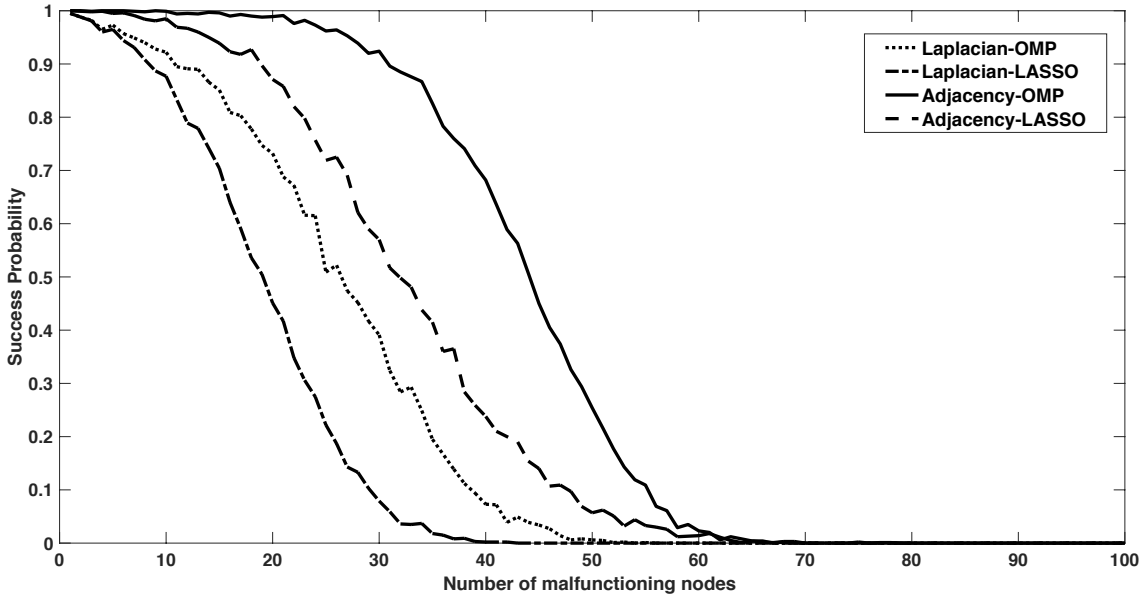



Figure 8.5: Success Probability vs Number of malfunctioning nodes

# 9. CONCLUSION

In this thesis, we explored the connections between coding theory, compressed sensing and spectral estimation for graph signals. We developed a Reed-Solomon style decoding algorithm for recovering graph signals that are $k$-sparse in the GFT domain from $2k$ aggregated samples. We then presented two real world applications namely MAC channel and anomaly detection. In the MAC channel problem, we developed a power efficient method for communication between the sensors and the base station. In the anomaly detection problem, we considered the noisy scenario where CS algorithms were employed to find out the malfunctioning nodes. An important thing to conclude from this work is that clever sampling strategies induce good codes. For example, here we applied aggregation sampling which in turn induced an RS code. Therefore, an open problem that comes out from this thesis is that efficient sampling strategies can be designed to induce less complex codes and subsequently less complex decoders like Low Density Parity Check(LDPC) can be designed for the spectral estimation problem. A further extension of this or a challenge will be to apply these in a noisy environment. Another open problem is the performance comparison of the Laplacian and adjacency matrix approach in the CS framework. It would be interesting to find out applications/graphs where each of these approaches would perform better than the other.

# REFERENCES


[1] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[2] M. W. Mahoney, M. Maggioni, and P. Drineas, "Tensor-CUR decompositions for tensor-based data," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 957–987, 2008.

[3] E. Acar and B. Yener, "Unsupervised multiway data analysis: A literature survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 1, pp. 6–20, 2009.

[4] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[5] D. L. Donoho and C. Grimes, "Hessian Eigenmaps: Locally linear embedding techniques for high-dimensional data," *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[6] S. Foucart and H. Rauhut, *A mathematical introduction to compressive sensing*, vol. 1. Birkhäuser Basel, 2013.

[7] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[8] M. A. Davenport and M. B. Wakin, "Analysis of orthogonal matching pursuit using the restricted isometry property," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4395–4401, 2010.

[9] "Graph connections," `http://www.touchgraph.com/facebook`, 2017.

[10] F. R. Chung, *Spectral graph theory*, vol. 92. American Mathematical Soc., 1997.

[11] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[12] M. Belkin and P. Niyogi, "Semi-supervised learning on manifolds," *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[13] M. Crovella and E. Kolaczyk, "Graph wavelets for spatial traffic analysis," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3, pp. 1848–1857, IEEE, 2003.

[14] M. Puschel and J. M. Moura, "Algebraic signal processing theory: Foundation and 1-D time," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3572–3585, 2008.

[15] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[16] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[17] F. Chung, "Laplacians and the cheeger inequality for directed graphs," *Annals of Combinatorics*, vol. 9, no. 1, pp. 1–19, 2005.

[18] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs: Frequency analysis.," *IEEE Trans. Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

[19] I. Pesenson, "Sampling in paley-wiener spaces on combinatorial graphs," *Transactions of the American Mathematical Society*, vol. 360, no. 10, pp. 5603–5627, 2008.

[20] J. Kovacevic and M. Puschel, "Algebraic signal processing theory: Sampling for infinite and finite 1-D space," *IEEE Transactions on Signal Processing*, vol. 58, no. 1, pp. 242–257, 2010.

[21] A. Anis, A. Gadde, and A. Ortega, "Towards a sampling theorem for signals on arbitrary graphs," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 3864–3868, IEEE, 2014.

[22] S. Chen, A. Sandryhaila, and J. Kovačević, "Sampling theory for graph signals," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 3392–3396, IEEE, 2015.

[23] S. Chen, R. Varma, A. Singh, and J. Kovacević, "Signal recovery on graphs: Random versus experimentally designed sampling," in *Sampling Theory and Applications (SampTA), 2015 International Conference on*, pp. 337–341, IEEE, 2015.

[24] G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst, "Random sampling of bandlimited signals on graphs," *Applied and Computational Harmonic Analysis*, 2016.

[25] S. Chen, R. Varma, A. Singh, and J. Kovačević, "Signal recovery on graphs: Fundamental limits of sampling strategies," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 539–554, 2016.

[26] R. Varma, S. Chen, and J. Kovačević, "Spectrum-blind signal recovery on graphs," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015 IEEE 6th International Workshop on*, pp. 81–84, IEEE, 2015.

[27] X. Wang, P. Liu, and Y. Gu, "Local-set-based graph signal reconstruction," *IEEE Transactions on Signal Processing*, vol. 63, no. 9, pp. 2432–2444, 2015.

[28] A. Krishnamuthy, J. Sharpnack, and A. Singh, "Recovering graph-structured activations using adaptive compressive measurements," in *Signals, Systems and Comput-*

*ers, 2013 Asilomar Conference on*, pp. 765–769, IEEE, 2013.

[29] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations," *IEEE Transactions on Signal Processing*, vol. 64, no. 7, pp. 1832–1843, 2016.

[30] T. K. Moon, "Error correction coding," *Mathematical Methods and Algorithms. Jhon Wiley and Son*, 2005.